

3/01-317 05

JC955 U.S. PRO
10/021042
12/19/01

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office

出 願 年 月 日

Date of Application:

2001年 3月30日

出 願 番 号

Application Number:

特願2001-097964

出 願 人

Applicant(s):

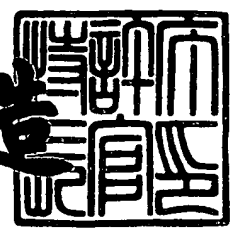
株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

2001年10月19日

特 許 庁 長 官
Commissioner,
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3092549

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

JC955 U.S. PTO
10/02/02
12/19/01

In re U.S. Patent Application of)
KAMINAGA et al.)
Application Number: To Be Assigned)
Filed: Concurrently Herewith)
For: ATTACK-RESISTANT IMPLEMENTATION METHOD)

Honorable Assistant Commissioner
for Patents
Washington, D.C. 20231

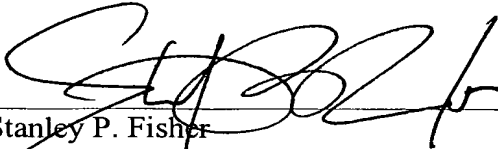
**REQUEST FOR PRIORITY
UNDER 35 U.S.C. § 119
AND THE INTERNATIONAL CONVENTION**

Sir:

In the matter of the above-captioned application for a United States patent, notice is hereby given that the Applicant claims the priority date of March 30, 2001, the filing date of Japanese patent application 2001-097964.

The certified copy of Japanese patent application 2001-097964 is being submitted herewith. Acknowledgment of receipt of the certified copy is respectfully requested in due course.

Respectfully submitted,


Stanley P. Fisher
Registration Number 24,344

REED SMITH HAZEL & THOMAS LLP
3110 Fairview Park Drive
Suite 1400
Falls Church, Virginia 22042
(703) 641-4200

JUAN CARLOS A. MARQUEZ
Registration No. 34,072

December 19, 2001

【書類名】 特許願

【整理番号】 H01003171A

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 19/00

【発明者】

【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所中央研究所内

【氏名】 神永 正博

【発明者】

【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所中央研究所内

【氏名】 遠藤 隆

【発明者】

【住所又は居所】 東京都国分寺市東恋ヶ窪一丁目 2 8 0 番地 株式会社日立製作所中央研究所内

【氏名】 渡邊 高志

【特許出願人】

【識別番号】 000005108

【氏名又は名称】 株式会社 日立製作所

【代理人】

【識別番号】 100075096

【弁理士】

【氏名又は名称】 作田 康夫

【電話番号】 03-3212-1111

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1
【物件名】 要約書 1
【プルーフの要否】 要

【書類名】明細書

【発明の名称】情報処理装置の演算方法および耐タンパー演算攪乱実装方式

【特許請求の範囲】

【請求項 1】

プログラムを格納するプログラム格納部およびデータを格納するデータ格納部を有する記憶手段と、演算処理装置と、前記演算処理装置で演算処理する対象となるデータを入力する手段および前記データの演算処理装置での演算処理結果を出力する手段とを有する情報処理装置内で 2 つの整数 K_1 、 K_2 に対して $F(K_1 + K_2, A) = F(K_1, A) \circ F(K_2, A)$ を満たす関数 F (ここで \circ は可換半群 S における二項演算を表す。 K は整数、 A は S の元を表す。) の値 $F(K, A)$ を求めるに際し、前記 K を m 個の整数の和 $K[0] + K[1] + \dots + K[m-1]$ に分解し、

前記 m 個の整数列 $0, 1, \dots, m-1$ の順序を置換 T で並べ替えた結果である $T(0), T(1), \dots, T(m-1)$ (これらの結果は前記整数列 $0, 1, \dots, m-1$ と一対一に対応するものとする。) を用いて

$$F(K, A) = F(K[T(0)], A) \circ F(K[T(1)], A) \circ \dots \circ F(K[T(m-1)], A) \quad \dots \quad (式 1)$$

の右辺中の項 $F(K[T(0)], A)$ から $F(K[T(m-1)], A)$ までを $F(K[T(0)], A), F(K[T(1)], A), \dots, F(K[T(m-1)], A)$ の順序で演算して $F(K, A)$ を算出することを特徴とする情報処理装置の演算方法。

【請求項 2】

前記置換 T による置換処理は置換前のデータから置換後のデータが予測できない情報源に基づいて行われるか又は疑似乱数に基づいて行われるものであり、かつ、前記 (式 1) の演算を行うたびにその置換処理を行うことを特徴とする請求項 1 記載の情報処理装置の演算方法。

【請求項 3】

前記 S は、整数の任意の整数 $N (N \geq 2)$ による剰余からなる集合に対して演算 \circ として法 N による乗算剰余演算 $A \circ B = A * B \bmod N$ を導入した可換半群であって、かつ、前記 F は $F(K, A) = A^K \bmod N$ (ここで A^K は A の K 乗を表す) であることを特徴とする請求項 1 記載の情報処理装置の演算方法。

【請求項 4】

前記情報処理装置をICカードに実装したことを特徴とする請求項 1 記載の情報処理装置の演算方法。

【請求項 5】

前記整数 K は $K[j] = u * ((2^t)^j) (0 \leq u \leq (2^t) - 1, t = 1, 2, \dots)$ の形に分割されることを特徴とする請求項 3 記載の情報処理装置の演算方法。

【請求項 6】

前記置換 T による置換処理は置換前のデータから置換後のデータが予測できない情報源に基づいて行われるか又は疑似乱数に基づいて行われるものであり、かつ、前記 (式 1) の演算を行うたびにその置換処理を行うことを特徴とする請求項 5 記載の情報処理装置の演算方法。

【請求項 7】

前記整数 K は $K[j] = u * ((2^t)^j) (0 \leq u \leq (2^t) - 1, t = 1, 2, \dots)$ の形に分割されることを特徴とする請求項 5 記載の情報処理装置の演算方法。

【請求項 8】

前記 S は、有限体 $GF(p)$ (p は素数) または $GF(2^n)$ (ここで n は 1 以上の整数) 上で定義される楕円曲線 E のモデル・ヴェイユ群であって、かつ、 $F(K, A) = KA$ であるものであり、ここで前記 A は前記楕円曲線 E 上の点を表す。前記 KA は K 個の A に対して演算 \circ を施したものを表す。即ち、前記 KA は前記 K が正のとき $A \circ A \circ A \cdots \circ A$ (K 個) を表し、前記 K が負のときは $(-A) \circ (-A) \circ (-A) \cdots \circ (-A)$ ($|K|$ 個) を表し、前記 K が 0 のときは前記 E 上の無限遠点 (the point at infinity) O (オー) を表し、前記 \circ は前記モデル・ヴェイユ群における和演算を表し、前記 $-A$ は前記 A の前記モデル・ヴェイユ群における逆元であることを特徴とする請求項 1 記載の情報処理装置の演算方法。

【請求項 9】

前記情報処理装置をICカードに実装したことを特徴とする請求項 8 記載の情報処理装置の演算方法。

【請求項 10】

前記整数 K は $K[j] = u * ((2^t)^j) (0 \leq u \leq (2^t) - 1, t = 1, 2, \dots)$ の形に分割

されることを特徴とする請求項 8 記載の情報処理装置の演算方法。

【請求項 1 1】

前記情報処理装置を IC カードに実装したことを特徴とする請求項 1 0 記載の情報処理装置の演算方法。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は情報処理装置の演算方法および耐タンパー演算攪乱実装方式に関し、特に高いセキュリティを持つ IC カードなどの耐タンパー情報処理装置等に関するものである。

【0 0 0 2】

【従来の技術】

IC カードは、勝手に書き換えることが許されないような個人情報の保持や、秘密情報である暗号鍵を用いたデータの暗号化や暗号文の復号化を行う装置である。IC カード自体は電源を持っておらず、IC カード用のリーダライタに差し込まれると、電源の供給を受け、動作可能となる。動作可能になると、リーダライタから送信されるコマンドを受信し、そのコマンドに従って、データの転送等の処理を行う。IC カードの一般的な解説は、オーム社出版電子情報通信学会編水沢順一著「IC カード」などにある。

IC カードの構成は、図 1 に示すように、カード 101 の上に、IC カード用チップ 102 を搭載したものである。図に示すように、一般に IC カードは、IS07816 の規格に定められた位置に供給電圧端子 Vcc、グランド端子 GND、リセット端子 RST、入出力端子 I/O、クロック端子 CLK を持ち、これらの端子を通して、リーダライタから電源の供給やリーダライタとのデータの通信を行う (W.Rankl and Effing : SMARTCARD HANDBOOK, John Wiley & Sons, 1997, pp.41 参照)。

IC カード用チップの構成は、基本的には通常のマイクロコンピュータと同じ構成である。その構成は、図 2 に示すように、中央処理装置 (CPU) 201、記憶装置 204、入出力 (I/O) ポート 207、コ・プロセッサ 202 からなる (コ・プロセッサはない場合もある)。CPU 201 は、論理演算や算術演算などを行う装置であり、記憶装

置204は、プログラムやデータを格納する装置である。入出力ポートは、リーダライタと通信を行う装置である。コ・プロセッサは、暗号処理そのもの、または、暗号処理に必要な演算を高速に行う装置であり、例えば、RSA暗号の剰余演算を行う為の特別な演算装置や、DES暗号のラウンド処理を行う暗号装置などがある。ICカード用プロセッサの中には、コ・プロセッサを持たないものも多くある。データバス203は、各装置を接続するバスである。

記憶装置204は、ROM(Read Only Memory)やRAM(Random Access Memory)、EEPROM(Electrical Erasable Programmable Read Only Memory)などからなる。ROMは、変更できないメモリであり、主にプログラムを格納するメモリである。RAMは自由に書き換えができるメモリであるが、電源の供給が中断されると、記憶している内容は消滅する。ICカードがリーダライタから抜かれると電源の供給が中断されるため、RAMの内容は、保持されなくなる。EEPROMは、電源の供給が中断されてもその内容を保持することができるメモリである。書き換える必要があり、ICカードがリーダライタから抜かれても、保持するデータを格納するために使われる。例えば、プリペイドカードでのプリペイドの度数などは、使用するたびに書き換えられ、かつリーダライタが抜かれてもデータを保持する必要があるため、EEPROMで保持される。

ICカードは、プログラムや重要な情報がICカード用チップの中に密閉されているため、重要な情報を格納したり、カードの中で暗号処理を行うために用いられる。従来、ICカードでの暗号を解読する難しさは、暗号アルゴリズムの解読の困難さと同じと考えられていた。しかし、ICカードが暗号処理を行っている時の消費電力を観測し、解析することにより、暗号アルゴリズムの解読より容易に暗号処理の内容や暗号鍵が推定される可能性が示唆されている。消費電力は、リーダライタから供給されている電力を測定することにより観測することができ、この攻撃法の詳細は、John Wiley & sons社 W.Rankl & W.Effing著「Smart Card Handbook」の8.5.1.1 Passive protective mechanisms(263ページ)にこのような危険性が記載されている。

ICカード用チップを構成しているCMOSは、出力状態が1から0あるいは0から1に変わった時に電力を消費する。特に、データバス203においては、バスドラ

イバーの電力や、配線及び、配線に接続されているトランジスタの静電容量のため、バスの値が1から0あるいは0から1に変わると、大きな電力が流れる。そのため、消費電力を観測すれば、ICカード用チップの中で、何が動作しているか分かる可能性がある。

図3は、ICカード用チップの1サイクルでの消費電力の波形を示したものである。処理しているデータに依存して、電力波形が301や302のように異なる。このような差は、バス203を流れるデータや中央演算装置201で処理しているデータに依存して生じる。

コ・プロセッサ202は、CPUと並列に、例えば、512ビットの剰余演算を行うことができる。そのため、CPUの消費電力とは異なった消費電力波形の長時間の観測が可能である。その特徴的な波形を観測することにより、コ・プロセッサの動作回数を容易に測定することができる。コ・プロセッサの動作回数が暗号鍵と何らかの関係があるならば、コ・プロセッサの動作回数から暗号鍵を推定できる可能性がある。

また、コ・プロセッサでの演算内容が、暗号鍵に依存した偏りを持つと、その偏りが消費電力として現れ、暗号鍵が推定される可能性がある。

CPUでも同様の事情が存在する。暗号鍵のビット値は決まっているため、処理するデータを変更して、消費電力を観測することにより、暗号鍵のビット値の影響が観測できる可能性がある。これらの消費電力の波形を統計的に処理することにより、暗号鍵を推定できる可能性がある。

【 0 0 0 3 】

【発明が解決しようとしている課題】

本発明の課題は、ICカード用チップでのデータ処理と消費電力との関連性を減らすことである。消費電力とチップの処理との関連性が減れば、観測した消費電力の波形からICカードチップ内での処理や暗号鍵の推測が困難になる。本発明の着眼点は、ICカードチップでの処理順序をアタッカーに推定されないように変更することにより、消費電力の波形から、処理や暗号鍵の推測を困難にするものである。

【 0 0 0 4 】

【課題を解決するための手段】

ICカードチップに代表される耐タンパー装置は、プログラムを格納するプログラム格納部、データを保存するデータ格納部を持つ記憶装置と、プログラムに従って所定の処理を実行しデータ処理を行う中央演算装置(CPU)を持ち、プログラムは、CPUに実行の指示を与える処理命令から構成される一つ以上のデータ処理手段からなる情報処理装置として捉えることができる。

本発明において、処理しているデータとICカード用チップの消費電力の関連性を攪乱する方法は、演算の順序を、本来のものとは異なるものに変化させるものである。一般に、処理の順序が異なれば、処理結果も変化するが、本発明の適用は、演算 $F(K, A)$ が、 K_1, K_2 に対し、 $F(K_1 + K_2, A) = F(K_1, A) \circ F(K_2, A)$ を満たすことを仮定する。この性質を繰り返し用いることにより、 K を m 個の整数の和 $K[0] + K[1] + \dots + K[m-1]$ に分解したとき、

$$F(K, A) = F(K[0], A) \circ F(K[1], A) \circ \dots \circ F(K[m-1], A)$$

となることがわかる。この性質を満たす F として、 $F(K, A) = A^{k \bmod N}$ （この場合、演算 $A \circ B$ は、 $A * B \bmod N$ である）や、 $F(k, A) = kA$ （但し、 A は楕円曲線上の点、演算 \circ は、楕円曲線上の点の和演算で、 kA は、 A を k 個加える演算を表す）がある。

本発明では、この性質を利用して、各 $F(K[0], A), F(K[1], A), \dots, F(K[m-1], A)$ を計算し、これを用いて、 $F(K, A) = F(K[0], A) \circ F(K[1], A) \circ \dots \circ F(K[m-1], A)$ を順番に計算する代わりに、該 $F(K, A)$ の計算を実行するたびに、 $0, 1, 2, \dots, m-1$ を置換 T によって並べ替えた結果 $T(0), T(1), \dots, T(m-1)$ に基づいて、演算順序を変更し、 $F(K, A) = F(K[T(0)], A) \circ F(K[T(1)], A) \circ \dots \circ F(K[T(m-1)], A)$ という処理順序で計算する。

この処理によって、観測される電力波形に現れる部分情報からだけでは元のデータ特定することが困難になる。統計処理を行う場合は、ランダムな波形を平均化することと同じことになり、波形の特徴は消滅するため、さらに効果は顕著である。この処理順序の変更が、予測できない情報源に基づいてなされれば、さらに攪乱効果が高まる。

本発明は、特に、RSA暗号での、乗算剰余演算や、べき乗剰余演算及び、楕円

曲線暗号での、定義体上での乗算や、除算、楕円曲線上の点のスカラー倍などの処理の情報隠蔽に利用することができる。

【0005】

【発明の実施の形態】

本実施例では、公開鍵暗号（非対称鍵暗号）の代表例である、RSA暗号、楕円曲線暗号を例にとる。これは、他の暗号にも用いることができる。RSA暗号については、岡本栄司著「暗号理論入門」（共立出版）や、A.J.Menezes, P.C. van Oorschot, S. A. Vanstone著 Handbook of Applied Cryptography, (CRC-Press)などに詳しく記載されている。楕円曲線暗号については、考案者の一人によって書かれたN. コブリツ著（櫻井幸一訳）「数論アルゴリズムと楕円暗号理論入門」（シュプリンガーフェアラーク東京）、楕円曲線上の演算については、I.H. シルバーマン・J. テイト著「楕円曲線論入門」（シュプリンガーフェアラーク東京）、又、群、環、体等の代数系については、松坂和夫著「代数系入門」（岩波書店）に詳しい説明がある。

一般に公開鍵暗号（非対称鍵暗号）においては、秘密鍵情報が、公開鍵の中に含まれているが、公開鍵から秘密鍵情報を取り出すことが、計算時間の点で著しく現実性を欠くということ（計算量的安全性）を根拠にして暗号が構成されている。計算量的安全性を持つ問題の代表的なものとして、素因数分解と、群上の離散対数問題が挙げられる。前者を利用したものがRSA暗号であり、後者を楕円曲線上の群に適用して利用しているものが、楕円曲線暗号である。

簡単にRSA暗号を説明する。

RSA暗号では、大きな素数、例えば512ビットの2つの素数 p, q の積 $N = pq$ と N と互いに素な数 e （ICカードでは、3や、65537が用いられることが多い）をとり、これを公開鍵として公開鍵簿に登録する。このとき、この公開鍵の持ち主Aに送信者Bは、1以上 $N-1$ 以下の数で表現されたデータ（平文） M を、

$$y = M^e \bmod N$$

として暗号化して送信する。ここで、 M^e は M の e 乗を表す記号とする。

この暗号文 C を受け取ったAは、 $xe \bmod (p-1)(q-1) = 1$ となる秘密鍵 x を用いて

$$S = y^x \bmod N$$

を計算する。ここで、 $(p-1)(q-1)$ は、 N のオイラー関数の値 $\phi(N)$ である。これは、 N と互いに素な自然数の個数に等しい。オイラーの定理によれば、

$$y^{((p-1)(q-1))} \bmod N = 1$$

が成り立ち、一方で、 $xe = 1 + k(p-1)(q-1)$ (k は整数) と書くことができるので、

$$\begin{aligned} y^x \bmod N &= M^{(xe)} \bmod N \\ &= M^{(1+k(p-1)(q-1))} \bmod N \\ &= M * M^{(k(p-1)(q-1))} \bmod N \\ &= M \end{aligned}$$

が成り立つ。従って、 $y^x \bmod N$ を計算することで、 A は、送信者 B の平文 M を復号することができる。この際、秘密鍵 x を計算するのに、 N そのものではなく、 N の素因数 p, q が用いられていることは、極めて重要である。現在のところ、 N の素因数分解を介さないで、 x を計算する方法は知られていない。大きな素数の積を因数分解することは、現実的でない時間が必要であるので、 N を公開しても、 A の秘密鍵は安全である。

上記RSA暗号の計算は、整数の N を法とする剰余全体 $Z(N)$ に、法 N における積をその演算とした半群(semigroup)の上で定義することができる。一般に、 N が素数でない場合は、 $Z(N)$ は、積に関する逆元を持たないので、群にはならないことに注意する。

RSA暗号の暗号化/復号化操作で用いられる演算は、べき乗剰余計算と呼ばれ、その計算機上での実装は、通常、図4のアルゴリズムで行われる。この実装方式は、アディション・チェイン方式(additional chain method)と呼ばれるものである。ここに示すのは、 $y^x \bmod N$ の計算を、秘密鍵 x のビットを2ビット毎に区切り、上位から読んでいき、それが、00,01,10,11のいずれであるかに応じて、 $A[0] = 1, A[1] = y, A[2] = y^2 \bmod N, A[3] = y^3 \bmod N$ を対応させ、乗算剰余計算を行うことによって実現したものである。勿論、2ビット毎に区切ることは説明の便宜のためであり、実際には1ビット、3ビット、4ビットをま

とめて計算することもある。その際も考え方は全く同じである。

この処理を図4に示す。まず、2ビット処理用のテーブル0401を用意する。Sを1に初期化し0411、

4乗処理0402に移る。この4乗処理0402は、xのビットと無関係に行われるが、その次に行われる乗算剰余計算においては、xのビット（2ビット分を指す）値に応じて、条件分岐0403、0404、0405、0406が行われ、それぞれ、0407、0408、0409、0410の乗算剰余計算が行われる。この際、異なるのは、テーブル0401の値： $A[0]$ 、 $A[1]$ 、 $A[2]$ 、 $A[3]$ である。一般に乗算剰余計算は処理が重く（その為、多くのICカードでは、乗算剰余処理専用のコ・プロセッサが用いられている）、その際、発生する電力は非常に大きい。特に、多桁計算の際に、 $A[0]$ 、 $A[1]$ 、 $A[2]$ 、 $A[3]$ のうち、どれが処理されているかがわかることがある。簡単のため、16ビットの計算を考えると、例えば、 $y = 58981$ 、 $N = 59989 (= 239 \times 251)$ として、 $A[0]$ 、 $A[1]$ 、 $A[2]$ 、 $A[3]$ を2進数表現すると、

$A[0] = 0000000000000001$

$A[1] = 0011001010011000$

$A[2] = 1011001011001110$

$A[3] = 1001111110010101$

のようなビット列になり、その違いに対応して、異なる電力波形が生ずる。このような違いから波形パターンが4種類に分類できれば、後は、その4つの順列のパターン数 $4! = 24$ 通りを試せば、秘密鍵のビットパターンを見つけることができる。法nのビット数が増えた場合も、同様である。

この攻撃方法は、特にNのビット数が増加したときに、著しい効果を発揮する。例えば、Nが2048ビットの場合、事実上因数分解を行うのは不可能であるが、オシロスコープを用いてチップの消費電力を見ることができれば、x（2000ビット程度）の値を知るには、2000ビット程度の波形の塊（2ビット毎に区切っていれば、約1000個の塊）を4通りに分類した後、その4通りそれぞれについて、別の計算機を用いて、べき乗剰余計算を実行し、チップから出力される結果と比較して、一致しているものを探し出せばよい。これは高々24通りにすぎない。

次に、上記のアディション・チェイン方式とは異なるRSA暗号のもう一つの実

装方法について述べる。これを図5～図7に示す。なお、図5のフローチャートの下端は図6のフローチャートの上端に連続するものである。この実装方式は、マイクロコンピュータのRAMの容量が十分大きい場合に可能なものである。ここでは、1024ビットのべき乗剰余計算を行う場合を想定する。以下、 x の二進数表示を $(x[0]x[1]\cdots x[511])$ と書く。ここで、 $x[j]$ は、2ビットブロックで、00, 01, 10, 11のいずれかに等しいものとする。

はじめに、 $B[j] = y^*(4^{(511-j)}) \bmod N$ ($j = 0, 1, \dots, 511$) テーブル作成を行う。 $S=y$ に初期化を行い0501、カウンタ j を512にセットする0502。この512という値は、1024ビットを2で割ったものである。(ここでは、2ビットずつ処理しているので、この値であるが、もし、4ビット処理であれば、 $1024/4 = 256$ となる。その他の場合(2のべき乗のウィンドウ幅の場合)も同様である。)

次に、条件分岐処理0503を実行する。該条件分岐処理は、終了条件を判定するものである。該終了条件が満たされていなければ、0504にて、 S をRAMに格納する。以下、カウンタ j に対する S の値を $B[j-1]$ と書く。この場合、 $B[0], B[1], \dots, B[511]$ を互いにRAM上で重複がないように格納する。例えば、図7のように1024ビット(128バイト)毎に連続的に配置する。ここでは、2バイト単位でアドレスが割り振られているものとする。このとき、データサイズは、1024ビット=128バイトであるから、128バイト毎のアドレスにテーブルを配置する。重複がないことはいうまでもない。格納後、0505にて、 S を法 N のもとで4乗する。4乗計算後、0506にてカウンタ j をデクリメントし、条件分岐処理0503に戻る。以下、終了条件が満たされるまで、この操作を512回繰り返す。この操作によって、カウンタ $j = 512$ に対しては、 $B[511]$ は、 y となり、511に対しては、 $B[510]$ は、 $y^*4 \bmod N$ となり、以下同様に、カウンタ j に対しては、 $B[j-1] = y^*(4^{(512-j)}) \bmod N$ が生成される。上記の操作は、データサイズのみ依存し、指数 x それぞれのビットには依存しない。

条件分岐処理0503の終了条件が満たされた場合、0507にて、 S を1に初期化する。又、0508にて、カウンタ j を0に設定する。条件分岐処理0509にて、もし、 j が512になっていれば、終了する0519。もし、終了条件が満足されていないならば、 $x[j]$ の値を調べ0510, 0511, 0512, 0513、それぞれの条件分岐に対して、05

14, 0515, 0516, 0517の処理を行う。その後、0518にて、カウンタをインクリメントし、条件分岐処理0509に戻る。この繰り返しは、512回実行される。

この処理で、正しい結果が得られる理由は、0514, 0515, 0516, 0517の処理において、Sに、

$$C[j] = B[j]^x[j] \bmod N \quad (j = 0, 1, 2, \dots, 511)$$

を乗じているため、全ての処理が終わった段階で、

$$\begin{aligned} S &= C[0] * C[1] * \dots * C[511] \bmod N \\ &= (y^{(x[0] * 4^{511})}) * (y^{(x[0] * 4^{510})}) * \dots * (y^{x[0]}) \bmod N \\ &= y^{(x[0] * 4^{511} + x[1] * 4^{510} + \dots + x[511])} \bmod N \\ &= y^x \bmod N \end{aligned}$$

となるためである。

この処理を用いた場合でも、電力解析により鍵情報が漏洩する可能性がある。理由は、通常のアディション・チェイン方式の場合と同様である。

次に楕円曲線暗号について、簡単に説明する。

楕円曲線とは、体Fの上で定義された、3次多項式の零点集合であり、Fの標数が2でない場合は、

$$y^2 = x^3 + ax^2 + bx + c$$

という標準形を持つ。標数が2の体の上では、

$$y^2 + cy = x^3 + ax + b \quad \text{又は、}$$

$$y^2 + xy = x^3 + ax + b$$

という標準形を持つ曲線である。(いずれの場合も、後に説明する無限遠点(the point at infinity)0を含めて考える)。楕円曲線の形状は、図8のようなものになる。図8は、 $y^2 = x^3 - 3x$ を示したものであるが、楕円曲線の形状は、係数によって大きく変化する。但し、通常、楕円曲線暗号では、右辺の三次式が重根を持たないという条件を必要とする。又、図8は、実数体上の図であり、実数体以外の体の上での楕円曲線の形状は、異なる場合がある。

本発明において、標数が2であるか否かは、本質的ではないので、以下、簡単のため、標数が2でない場合について説明する。又、暗号で必要なのは、有限体の場合のみであるので、その場合に限って説明する。有限個の元からなる体を有

限体又はガロア体といい、その構造はよく知られている。その最も単純な構成法は以下の通りである。

まず、素数 p を法とする整数環の剰余環 $Z(p)$ を考える。 $Z(p)$ においては、0以外の元は逆を持つので、体の構造を持っている。これを素体といい、 $GF(p)$ と書く。これが最も原始的な有限体の例である。

次に、 $GF(p)$ の元を係数に持つ多項式 $f(X)$ を考え、その零点のうち、 $GF(p)$ に含まれないものを $GF(p)$ に添加することによって、新しい体を構成することができる。これを、 $GF(p)$ の有限次代数拡大体という。 $GF(p)$ の有限次代数拡大体の元の個数は、 p のべきになっていることが知られている。その元の個数を q と書くとき、有限次代数拡大体を $GF(q)$ などと表示することがある。

楕円曲線上の点の間には、演算を定めることができる。図9に示すように、楕円曲線上の二つの点、 P, Q があるとき、この二点を通る直線を引き ($P=Q$ のときは接線を引き)、この直線が再び楕円曲線と交わる点 S を x 軸に関して対称に折り返した点は、曲線の対称性から、再び楕円曲線上の点となる。この点を $P+Q$ と書き、 P と Q の「和」と定義する。交わる点がない場合は、架空の点として無限遠点 (the point at infinity) というものを考え、この架空の点 (virtual point) で交わっているものとみなす。無限遠点 (the point at infinity) を 0 と書く。点データの表現方法は、多々あるが、ここでは、例えば、射影座標 (projective coordinate) を用いる場合を考える。通常座標から射影座標への変換の方法は、例えば、次のようにする。まず、通常の二次元平面から原点 (origin) を除去した集合を H とする。 H の点の座標 (x, y) に対して、 x を X/Z , y を Y/Z と置換えて、これを (X, Y, Z) という3次元の点と考える。すると、0でないスカラー c に対し、 (cX, cY, cZ) は、 H 上の同一の点に対応する。逆に言えば、3次元空間において、 (cX, cY, cZ) (c は0でない) と書ける点を全て同一視したものを H とみなすことができる。また、射影空間では、 $(X, Y, 0)$ という形の点 (正確には、その同値類 (equivalent class)) が無限遠点 (the point at infinity) 0 に対応する。(一般に、楕円曲線 (elliptic curve) を含む代数多様体 (algebraic variety) は、射影空間の上で考えることが多い。) また、楕円曲線上の点 P と x 軸に関して対称な位置にある点を P の逆元といい、 $-P$ で表す。 $G(E/Fq)$ における一点 P を k 個加えたものを、 kP 、 $-P$

を k 個加えたものを $-kP$ と書いて、 P のスカラー倍という。これらの座標は、 P, Q の座標の有理式で表すことができ、従って、一般の体の上でこの演算を考えることができる。この「加法」は、通常の加法と同様に、結合法則、交換法則が成立し、この加法に関して、無限遠点(the point at infinity) O は、通常の数での演算と同様にゼロの役割を果たし、 $-P$ は、 P と加えると、 O になる。これは楕円曲線上の加法演算が、可換群（アーベル群）の構造を持つことを示している。これをモデル・ヴェイユ群ということがある。楕円曲線 E 、定義体 $GF(q)$ を固定したときのモデル・ヴェイユ群を、 $G(E/GF(q))$ と書くことがある。 $G(E/GF(q))$ の構造は非常に単純で、巡回群か、または二つの巡回群の直積と同型になることが知られている。

一般に、 $kP=Q$ の値がわかって、逆に k の値を知るのは計算量が膨大であるため容易でない。これを楕円曲線上の離散対数問題という。楕円曲線暗号は、楕円曲線上の離散対数問題が困難であることに基づいている。

楕円曲線を利用した暗号方式には種々のものがあるが、ここでは、特に、楕円ElGamal方式を説明する。

楕円曲線 E とその上の一点（一般に大きな位数を持つ点。ベースポイントと呼ぶ） P が公開されているものとする。

A氏が、B氏に秘密情報 M （楕円曲線上の点で表現する。平文（暗号文）の楕円曲線上の埋め込みについては、N.コブリッツ、櫻井幸一訳「数論アルゴリズムと楕円暗号入門」シュプリンガーフェアラーク p.253に説明されている）を送信することを考える。

STEP 1. 受信者B氏は、正整数 $x[B]$ を選び、これを秘密鍵として保持し、

$$Y[B] = X[B]P$$

を公開鍵簿に登録する。

STEP 2. 送信者 A氏は、乱数 r を用いて、

$$C1 = rP$$

$$C2 = M + rY[B]$$

をB氏に送信する。

STEP 3. 受信者B氏は、 $C1, C2$ を受け取り、自分の秘密鍵 $X[B]$ を用いて、

$$C2 \times [B] C1 = M$$

としてMを復元する。

楕円ElGamal暗号に限らず、楕円曲線暗号においては、楕円曲線上の点のスカラー倍を計算する必要がある。

楕円曲線上の点のスカラー倍を求めるアルゴリズムは、べき乗剰余計算のアルゴリズムに非常に類似している。 kP (k は正整数)を計算する標準的なアルゴリズムをべき乗剰余計算と同様に2ビットづつまとめて処理する場合について示したものを、図10に示す。構成が全く同じであることがわかる。この処理方式もアディション・チェーン方式と呼ばれる。(但し、計算機に実装する際には、個々の計算はRSAとは大幅に違ったものになる。)

まず、2ビットの処理を一度に行う為に、受信した点Pに対するルックアップテーブルを作る。べき乗剰余演算においては、mod Nでの0乗、1乗、2乗、3乗に対応して、 $P[0]=0$ (無限遠点(the point at infinity))、 $P[1]=P$ 、 $P[2]=2P$ 、 $P[3]=3P$ を用意する(0801)。次に、計算用の点の値を初期化する0802。次に終了条件を判定し0803、終了条件が満たされていれば終了する0813。終了していなければ、Sを4倍する処理0804を実行した後、kのビットの値(2ビット毎)を見て条件分岐して0805、0806、0807、0808これらに対応する点 $P[0]=0$ 、 $P[1]$ 、 $P[2]$ 、 $P[3]$ を加える0809、0810、0811、0812。この処理をkのビットが尽きるまで続けることにより、 kP を計算することができる。この計算はkの上位から2ビットづつ区切って見て計算する方式である。べき乗剰余計算と、数学的には同一の構造をしていることがわかる。後に再び説明するが、RSAにおけるべき乗剰余演算、楕円曲線上の加法演算は、それぞれ、 $Z(N)$ 、 $G(E/GF(q))$ という代数系の上で行われる演算と考えることができ、これらを、より一般の代数系に拡張することは極めて自然なことである。その際の計算機演算の方法も、概ね、ここで述べたアルゴリズムにより処理される。

一方で、マイクロコンピュータが内部のプログラムを実行する際、動作時の内部消費電力が漏洩する可能性があるため、このプロセスをマイクロコンピュータで実現する場合、秘密鍵の処理が漏洩し、危険に曝されることになる。例えば、kのビット(この例では2ビット毎)の違いにより、分岐が行われるので、もし

も、その処理が消費電力の差として現れれば、電力波形から k のビットを特定することができる可能性がある。

楕円曲線暗号の場合も、RSAの場合と同様に、アディション・チェイン方式において、あらかじめ $4^m p$ の形の点を計算して、ルックアップテーブルとしてメモリ上に置くことができる。楕円曲線暗号においては、この各点の値は、入力に全く依存しないという点がRSAと比較して実装上極めて有利な点である。RSAの場合、入力毎にテーブルを作成しなければならないだけでなく、これらテーブル値をRAMに置かなければならない。楕円曲線暗号では、テーブル値が、入力非依存であるので、ROMや、EEPROMのような不揮発性のメモリにあらかじめ値を用意しておくことができる。これは、計算時間の節約という点でも大きな利点である。

このようなルックアップテーブルを用いたアディション・チェイン方式のフローを図11、12に示す。なお、図11のフローチャートの下端は図12のフローチャートの上端に連続するものである。

ここでは、160ビットのスカラー倍計算を行う場合を想定する。尚、楕円曲線暗号での鍵長160ビットは、RSA暗号の鍵長1024ビットの強度に相当することが知られている。以下、 k の二進数表示を $(k[0]k[1]\cdots k[79])$ と書く。ここで、 $x[j]$ は、2ビットブロックで、00, 01, 10, 11のいずれかに等しいものとする。

テーブル $B[j] = (4^{(79-j)})P$ ($j = 0, 1, \dots, 79$)は、事前に計算し、EEPROMに格納しておくこともできるし、毎回計算することもできる。事前に計算してEEPROMに格納しておく場合は、図11の処理は不要である。ここでは、現実的には無駄な処理であるが、RSAの場合と対比しやすくするために、テーブルを毎回計算する場合を考える。

まず、 $S=P$ に初期化を行い0901、カウンタ j を160にセットする0902。この80という値は、160ビットを2で割ったものである。（ここでは、2ビットずつ処理しているので、この値であるが、もし、4ビット処理であれば、 $160/4 = 40$ となる。その他の場合（2のべき乗のウィンドウ幅の場合）も同様である。）

次に、条件分岐処理0903を実行する。該条件分岐処理は、終了条件を判定するものである。該終了条件が満たされていなければ、0904にて、 S をRAMに格納す

る。以下、カウンタ j に対する S の値を $B[j-1]$ と書く。この場合、 $B[0]$, $B[1]$, ..., $B[79]$ を互いに RAM 上で重複がないように格納する。格納方法の例は、RSA 暗号に対する配置図 7 において、128 バイトの部分をも 10 バイトに置換えることによって得られる。格納後、0905 にて、 S を楕円曲線上の演算の意味で 4 倍する。4 倍計算後、0906 にてカウンタ j をデクリメントし、条件分岐処理 0903 に戻る。以下、終了条件が満たされるまで、この操作を 80 回繰り返す。この操作によって、カウンタ $j = 80$ に対して、 $B[79]$ は、 P となり、79 に対して、 $B[78]$ は、 $4P$ となり、以下同様に、カウンタ j に対しては、 $B[j-1] = (4^{79-j})P$ が生成される。

条件分岐処理 0903 の終了条件が満たされた場合、0907 にて、 S を 0 (無限遠点 (the point at infinity)) に初期化する。又、0908 にて、カウンタ j を 0 に設定する。条件分岐処理 0909 にて、もし、 j が 80 になっていれば、終了する 0919。もし、終了条件が満足されていないならば、 $k[j]$ の値を調べ 0910, 0911, 0912, 0913、それぞれの条件分岐に対して、0914, 0915, 0916, 0917 の処理を行う。その後、0918 にて、カウンタをインクリメントし、条件分岐処理 0909 に戻る。この操作が、80 回繰り返される。

この処理で、正しい結果が得られる理由は、0914, 0915, 0916, 0917 の処理において、 S に、

$$C[j] = k[j] * B[j] \quad (j = 0, 1, 2, \dots, 79)$$

を (楕円曲線上の和の意味で) 加算しているため、全ての処理が終わった段階で、

$$\begin{aligned} S &= C[0] + C[1] + \dots + C[79] \\ &= k[0] * (4^{79})P + k[1] * (4^{78})P + \dots + k[79]P \\ &= (k[0] * 4^{79} + k[1] * 4^{78} + \dots + k[79])P \\ &= kP \end{aligned}$$

となるためである。

上記の楕円曲線上のスカラー倍演算を実現する際には、異なる座標系を用いることがあり、それに応じてマイクロコンピュータ内部での点の表現が異なる場合がある。しかし、代数学的には、双有理変換で移りあう曲線は全て同じモデル・ヴェイユ群の構造を与えるため、上記のアルゴリズムは、その本質において、

全く同一でよいという点に注意しておく。

以上の諸概念を踏まえた上で、図 1 3 ～ 図 1 6 を用いて実施例を説明する。なお、図 1 3 ～ 図 1 5 のフローチャートはこの図の順番に連続した一連のものである。本実施例は、1024ビットのRSA暗号処理（べき乗剰余演算） $S = y^x \bmod N$ を実行するためのものである。

図 1 3 は、計算に必要なテーブルを作成する処理を示したものである。まず、 S を y に初期化し1001、カウンタ j を512に設定する1002。次に、条件分岐処理1003にて、終了条件の判定を行う。次に、 S をRAMに格納する1004。（以下、この値をカウンタ j に対して、 $B[j]$ と書くとき、図 5 ～ 図 7 における処理と同様に、各 $B[j]$ は、互いにメモリ上で重複しないように配置していくものとする。）次に、 S を法 N の下で4乗し1005、カウンタ j をデクリメントし1006、再び条件分岐処理1003に戻る。この操作は、512回繰り返され、図 1 4 の処理1007に移る。1007では、再びカウンタ j に関する条件判定を行う。これは終了判定であり、 j が512でなければ、1008の処理に進む。1008では、図 1 6 に示すようなランダム置換用のEEPROM領域PERM_tmpから j 番目の値 $V(j)$ をリードする。図 1 6 に示すように、 $V(j)$ には、あらかじめ0, 1, 2, ..., 511をランダムに並べ替えたものが配置されているものとする。次に、1009にて、 $V(j)$ を、 $V(j) = ((17 * (V(j) + 1) \bmod 513) - 1)$ によって書き換える（本実施例では、 j 毎に書き換えているが、EEPROMによっては、ページ単位での書き換えしか許されない場合があり、その場合は、ページサイズ分の $V(j)$ をRAMに格納しておき、一括して書き直すように実装すべきである）。この部分の操作を、より一般的な立場から説明する。

集合 $S(n) = \{1, 2, 3, \dots, n\}$ 上で定義された変換 $U(w) = a * w \bmod (n+1)$ を考える。ユークリッドの互除法から容易にわかるように、 a が $n+1$ と互いに素であるときは、 a は、法 $n+1$ に対して逆を持つから、このような a に対しては、変換 U は $S(n)$ から $S(n)$ への全単射(bijection)である。言い換えれば、 U は、1, 2, 3, ..., n を並べ替える変換である。従って、 $U(w) = 17 * w \bmod 513$ は、 $513 = 3 * 3 * 3 * 19$ と、17が互いに素であるから $\{1, 2, 3, \dots, 512\}$ 上の全単射である。従って、1009に現れる変換 $((17 * (w + 1) \bmod 513) - 1)$ は、 $\{0, 1, 2, \dots, 511\}$ を並べ替えることがわかる。勿論、17という数は、一例

であって、513と互いに素であれば、 $\{0, 1, 2, \dots, 511\}$ の置換を定義する。
又、置換を生成する方法は、これだけではないが、本発明の本質とは無関係であるので、説明は省略する。

1009の処理の後、1023でカウンタjをインクリメントする。この操作を512回繰り返し、図15における1010の操作に進む。1010では、再びSを1に初期化する。また、カウンタjを0に初期化する1011。次に、1012にて、カウンタjが512であるかどうかを判定し、条件が満たされていれば終了する1022。該終了条件が満たされていなければ、1013、1014、1015、1016の条件分岐処理を行う。この処理は、秘密鍵指数(secret exponent)xのV(j)番目の値を読んで条件分岐するものである。ここでは、カウンタの値jをそのまま使わず、ランダム置換されたV(j)の値を用いて処理を行うことに注意する。これらの条件分岐処理1013、1014、1015、1016に対応して、1017、1018、1019、1020の処理を実行する。1018、1019、1020の処理においては、V(j)の値に対応するB[V(j)]を用いて乗算剰余処理を実行する。1017、1018、1019、1020の処理を終えた後、カウンタjをインクリメントし1021、条件分岐処理1012に戻る。

この処理で正しい結果が得られる理由は以下の通りである。

1017、1018、1019、1020の処理において、Sに、

$$C[V(j)] = B[V(j)]^x[V(j)] \bmod N \quad (j = 0, 1, 2, \dots, 511)$$

を乗じているため、全ての処理が終わった段階で、

$$\begin{aligned} S &= C[V(0)] * C[V(1)] * \dots * C[V(511)] \bmod N \\ &= (y^{(x[V(0)] * 4^{(511-V(0))})} * (y^{(x[V(1)] * 4^{(512-V(1))})} * \dots * (y^{(x[V(0)] * 4^{(511-V(511))})}) \bmod N \\ &= y^{(x[V(0)] * 4^{(511-V(0))} + x[V(1)] * 4^{(512-V(1))} + \dots + x[V(0)] * 4^{(511-V(511))})} \bmod N \end{aligned}$$

となる。

ここで、写像Vの性質から、V(0), V(1), ..., V(511)は、0, 1, ..., 511を並べ替えたものであるから、上記の指数部：

$x[V(0)] * 4^{(511-V(0))} + x[V(1)] * 4^{(512-V(1))} + \dots + x[V(0)] * 4^{(511-V(511))}$
は、 $x[0] * 4^{511} + x[1] * 4^{510} + \dots + x[511]$ に等しい。従って、Sは、 $y^x \bmod$

Nに等しい。

本実施例の処理を実行している際に発生する消費電力は、通常の処理（図5～図7で示した処理）の場合とは、著しく異なる。図5～図7で示した処理では、上位ビットから順に実行されるため、ビット毎の消費電力の違いを判別することにより、秘密鍵を同定できる可能性がある。一方、本実施例の処理においては、処理されるビット位置が予測できない情報源により攪拌される。一般に、秘密鍵 x のビットは、0と1が平均して同数に近い値になるため、00, 01, 10, 11という並びも、ほぼ均等に出現するため、攪拌の効果が期待できる。これは、本発明請求項1, 2, 3, 5の実施例の一つである。

次に、楕円曲線暗号に本発明を適用することを考える。

楕円曲線上のスカラー倍処理に対しても、RSA暗号におけるべき乗剰余演算と同様の方法を適用することができる。

図17～20で示す実施例は、160ビットの楕円曲線暗号処理（楕円曲線 E 上の点 P のスカラー倍演算） $S = kP$ を実行するためのものである。楕円曲線は、どのようなガロア体の上で定義されていても、本実施例の構成に差はないので、以下、定義体及び、該定義体の上での演算について、個別に述べることはしない。

図17は、計算に必要なテーブルを作成する処理を示したものである。まず、 S をベースポイント P に初期化し1101、カウンタ j を80に設定する1102。次に、条件分岐処理1103にて、終了条件の判定を行う。次に、 S をRAMに格納する1104。（以下、この値をカウンタ j に対して、 $B[j]$ と書くとき、図5～図7における処理と同様に、各 $B[j]$ は、互いにメモリ上で重複しないように配置していくものとする。）次に、 S を楕円曲線 E の上で4倍し1105、カウンタ j をデクリメントし1106、再び条件分岐処理1103に戻る。この操作は、80回繰り返され、図18の処理1107に移る。1107では、再びカウンタ j に関する条件判定を行う。これは終了判定であり、 j が80でなければ、1108の処理に進む。1108では、図20に示すようなランダム置換用のEEPROM領域 $PERM_tmp$ から j 番目の値 $V(j)$ をリードする。図20に示すように、 $V(j)$ には、あらかじめ0, 1, 2, ..., 79をランダムに並べ替えたものが配置されているものとする。次に、1109にて、 $V(j)$ を、 $V(j) = ((7 * (V(j) + 1) \bmod 81) - 1)$ によって書き換える（本実施例では、 j 毎に書き換えているが、 E

EPROMによっては、ページ単位での書換えしか許されない場合があり、その場合は、ページサイズ分の $V(j)$ をRAMに格納しておき、一括して書き直すように実装するべきである)。この操作が、 $\{1, 2, 3, \dots, 79\}$ を並べ替えるものであることは、先にRSA暗号処理に対する本発明実施例で示したものと同様である。勿論、7という数は、一例であって、81と互いに素であれば、 $\{0, 1, 2, \dots, 81\}$ の置換を定義する。又、置換を生成する方法は、これだけではないが、本発明の本質とは無関係であるので、説明は省略する。

1109の処理の後、1123でカウンタ j をインクリメントする。この操作を80回繰り返し、図19における1110の操作に進む。1110では、再び S を0（無限遠点(the point at infinity)）に初期化する。また、カウンタ j を0に初期化する1111。次に、1112にて、カウンタ j が80であるかどうかを判定し、条件が満たされていれば終了する1122。該終了条件が満たされていなければ、1113、1114、1115、1116の条件分岐処理を行う。この処理は、スカラー k の $V(j)$ 番目の値を読んで条件分岐するものである。ここでは、カウンタの値 j をそのまま使わず、ランダム置換された $V(j)$ の値を用いて処理を行うことに注意する。これらの条件分岐処理1113、1114、1115、1116に対応して、1117、1118、1119、1120の処理を実行する。1118、1119、1120の処理においては、 $V(j)$ の値に対応する $B[V(j)]$ を用いて楕円曲線上の和の処理を実行する。1117、1118、1119、1120の処理を終えた後、カウンタ j をインクリメントし1121、条件分岐処理1112に戻る。

この処理で正しい結果が得られる理由は以下の通りである。

1117、1118、1119、1120の処理において、 S に、

$$C[V(j)] = k[V(j)] * B[V(j)] \bmod N \quad (j = 0, 1, 2, \dots, 79)$$

を楕円曲線上で加えているため、全ての処理が終わった段階で、

$$\begin{aligned} S &= C[V(0)] + C[V(1)] + \dots + C[V(511)] \\ &= (k[V(0)] * 4^{(79-V(0))P} + (k[V(1)] * 4^{(512-V(1))P} + \dots + (k[V(0)] * 4^{(511-V(511))P} \end{aligned}$$

$$\begin{aligned} &= (k[V(0)] * 4^{(79-V(0))} + k[V(1)] * 4^{(79-V(1))} + \dots + k[V(0)] * 4^{(79-V(79))}) \\ &P \end{aligned}$$

となる。

ここで、写像Vの性質から、 $V(0), V(1), \dots, V(79)$ は、 $0, 1, \dots, 79$ を並べ替えたものであるから、上記のスカラー部：

$$k[V(0)] \cdot 4^{(79-V(0))} + k[V(1)] \cdot 4^{(79-V(1))} + \dots + k[V(79)] \cdot 4^{(79-V(79))}$$

は、 $k[0] \cdot 4^{79} + k[1] \cdot 4^{78} + \dots + k[79]$ に等しい。従って、Sは、 kP に等しい。

本実施例の処理を実行している際に発生する消費電力は、通常の処理（図11、12で示した処理）の場合とは、著しく異なる。図11、12の処理では、上位ビットから順に実行されるため、ビット毎の消費電力の違いを判別することにより、秘密鍵を同定できる可能性がある。一方、本実施例の処理においては、処理されるビット位置が予測できない情報源により攪拌される。一般に、暗号処理で用いられるスカラーkのビットは、0と1が平均して同数に近い値になるため、00, 01, 10, 11という並びも、ほぼ均等に出現するため、攪拌の効果が期待できる。これは、本発明請求項1, 2, 4, 5の実施例の一つである。

これまで、スカラーは、通常の二進数展開表示を利用してきたが、符号を考慮した、別の表示方法を考えることもできる。しかし、スカラーの表示方法は本発明の本質には無関係である。この事情について、簡単に説明する。

まず、楕円曲線上の和の演算は、逆元を持つことに注意する。このことを利用して、

$$(2^m - 1)P = (2^m)P + (-P)$$

という変形ができることに注目する。

例えば、195は、二進数で、11000011と書けるが、これを、2ビットブロックに分割し、(11,00,00,11)と書く。下位から見ていくと、最初に11があるので、11の上位に1を加え、(11,00,01,11)とする。最上位にも11があるので、再び上位に1を加え、(01,11,00,01,11)とする。このように変換したときは、11を-1と読み替えて解釈する。すなわち、

$$\text{変換前：} 195 = 3 \cdot 4^3 + 0 \cdot 4^2 + 0 \cdot 4 + 3$$

$$\text{変換後：} 195 = 1 \cdot 4^4 + (-4^3) + 0 \cdot 4^2 + 1 \cdot 4 + (-1)$$

として表現する。これは、 $3 = 4 - 1$ と書き換えているのと同様である。これをスカラー倍演算に適用すると、

$$195P = 1*((4^4)P) - (4^3)P + 0*((4^2)P) + 1*4P - P$$

となる。一般に、このような変形を行えば、スカラーのビット長が増加する。例えば、上記の例では、8ビットのスカラーが、10ビットに増加している。このように、このスカラー表現を用いるとスカラーの長さは増加する。しかし、楕円曲線上では符号を変える操作が容易であることや、テーブルの大きさが制限されているなどの理由により、ICカード上に実装されることがある。

このような変形を施した場合に本発明を適用することを考える。以下、受信する点をP、スカラーkのサイズは、160ビットとし、実行する演算は、kPとする。

スカラーkを、上記のスカラー表現方法で表現したものをk'で表現する。この表現k'は毎回計算してもよい（実装上スカラーの表現を変更できない場合がある）が、ここでは、あらかじめ用意しておくことを考える。通常k'は、EEPROMに格納する。スカラーk'のサイズは、もとのkの表現よりも大きくなる場合もあるが、ここでは、簡単のため、160ビットのままであるものとしておく。

このような準備をした上で、以下の処理を行う。

まず、図21の処理に従って $P[j] = (4^j)P$ ($j = 0, 1, 2, \dots, 80$)を計算し、ルックアップテーブルを作成する。まず、SをベースポイントPに初期化し1201、カウンタjを80に設定する1202。次に、条件分岐処理1203にて、終了条件の判定を行う。次に、SをRAMに格納する1204。（以下、この値をカウンタjに対して、B[j]と書くとき、図5～図7に示した処理と同様に、各B[j]は、互いにメモリ上で重複しないように配置していくものとする。）次に、Sを楕円曲線Eの上で4倍し1205、カウンタjをデクリメントし1206、再び条件分岐処理1203に戻る。この操作は、80回繰り返され、図22に示した処理1207に移る。1207では、再びカウンタjに関する条件判定を行う。これは終了判定であり、jが80でなければ、1208の処理に進む。1208では、図20に示すようなランダム置換用のEEPROM領域PERM_tmpからj番目の値V(j)をリードする。図20に示すように、V(j)には、あらかじめ0, 1, 2, ..., 79をランダムに並べ替えたものが配置されているものとする。次に、1209にて、V(j)を、 $V(j) = ((7*(V(j)+1) \bmod 81) - 1)$ によって書き換える。この操作が、{1, 2, 3, ..., 79}を並べ替えるものであることは、先

にRSA暗号処理に対する本実施例で示したものと同様である。勿論、7という数は、一例であって、81と互いに素であれば、 $\{0, 1, 2, \dots, 81\}$ の置換を定義する。又、置換を生成する方法は、これだけではないが、本発明の本質とは無関係であるので、説明は省略する。

1209の処理の後、1223でカウンタjをインクリメントする。この操作を80回繰り返し、図23における1210の操作に進む。1210では、再びSを0（無限遠点(the point at infinity)）に初期化する。また、カウンタjを0に初期化する1211。次に、1212にて、カウンタjが80であるかどうかを判定し、条件が満たされていれば終了する1222。該終了条件が満たされていなければ、1213、1214、1215、1216の条件分岐処理を行う。この処理は、スカラー k' の $V(j)$ 番目の値を読んで条件分岐するものである。ここでは、カウンタの値jをそのまま使わず、ランダム置換された $V(j)$ の値を用いて処理を行うことに注意する。これらの条件分岐処理1213、1214、1215、1216に対応して、1217、1218、1219、1220の処理を実行する。1218、1219、1220の処理においては、 $V(j)$ の値に対応する $B[V(j)]$ を用いて楕円曲線上の和の処理を実行する。但し、1220では、図17～図20の例とは異なり、 $3*B[j]$ を加える代わりに、 $-B[j]$ を加える（ $B[j]$ を引く）。この点のみが、図17～図20の例と異なっている。1217、1218、1219、1220の処理を終えた後、カウンタjをインクリメントし1221、条件分岐処理1212に戻る。この処理で正しい結果が得られる理由は図17～図20の実施例の場合と同様である。ビットの攪拌効果も同様である。

上記の実施例においては、秘密鍵指数や、スカラーの全てのビットを攪拌しているが、攪拌を行うビット位置を制限することもできる。たとえば、上記の実施例において、スカラーの上位16ビット目から、80ビット目までに本発明を適用することもできる。これは、該当部分のみ取り出せば、スカラー倍処理と同等であるからである。

以上に見たように、べき乗剰余演算、楕円曲線上の加算に対する本発明の実施例の思想は抽象レベルでは全く同一のものであって、これらを、実際のインプリメントを超えて抽象化することは自然なことである。

【0006】

これまで示してきた法Nにおけるべき乗剰余演算や、楕円曲線上の加算における実施例において、積又は和の演算を○と書けば全く同様のフローチャートが構成できる。これらは、上記のべき乗剰余演算及び、楕円曲線上の点のスカラー倍を含み、さらに同様の代数構造を持つ処理に対しても有効である。

【 0 0 0 7 】

【発明の効果】

本発明の実施例によれば、ICカードチップにおいて、結果に影響を与えないように処理順序の変形を行うことにより、消費電力の波形から、処理や暗号鍵の推測を行うことの困難性を高めることが可能である。

【図面の簡単な説明】

【図 1】

一般的なICカードの外観を示す図。

【図 2】

一般的なマイクロコンピュータの構成を示す図。

【図 3】

ICカード用チップでの1サイクル分の消費電力波形の例を示す図。

【図 4】

アディション・チェイン方式を用いたべき乗剰余計算を示すフローチャート。

【図 5】

アディション・チェイン方式を用いたべき乗剰余計算を示すフローチャート(テーブル作成部)。

【図 6】

アディション・チェイン方式を用いたべき乗剰余計算を示すフローチャート(モジュラ乗算部)。

【図 7】

アディション・チェイン方式を用いたべき乗剰余計算のためのテーブルのメモリ配置を示す図。

【図 8】

楕円曲線の形状を示す図。

【図 9】

楕円曲線上の加法を説明するための図。

【図 1 0】

アディション・チェイン方式を用いたスカラー倍計算を示すフローチャート。

【図 1 1】

楕円曲線上の点のスカラー倍計算部計算を示すフローチャート(テーブル作成部)。

【図 1 2】

楕円曲線上の点のスカラー倍計算部計算を示すフローチャート(スカラー依存処理)。

【図 1 3】

本発明実施例の説明図であって、RSA暗号処理のフローチャート(テーブル作成部)。

【図 1 4】

本発明実施例の説明図であって、RSA暗号処理のフローチャート(ランダム置換生成部)。

【図 1 5】

本発明実施例の説明図であって、RSA暗号処理のフローチャート(モジュラ乗算部)。

【図 1 6】

本発明実施例の説明図であって、RSA暗号処理に関連する置換テーブルを示す図。

【図 1 7】

楕円暗号処理の第 1 の実施例のフローチャート(テーブル作成部)。

【図 1 8】

楕円暗号処理の第 1 の実施例のフローチャート(ランダム置換生成部)。

【図 1 9】

楕円暗号処理の第 1 の実施例のフローチャート(スカラー依存処理)。

【図 2 0】

楢円暗号処理の第 1 の実施例の置換テーブルを示す図。

【図 2 1】

楢円暗号処理の第 2 の実施例のフローチャート(テーブル作成部)。

【図 2 2】

楢円暗号処理の第 2 の実施例のフローチャート(ランダム置換生成部)。

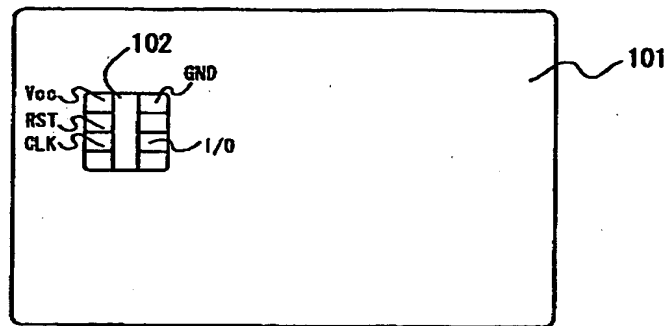
【図 2 3】

楢円暗号処理の第 2 の実施例のフローチャート(スカラー依存処理)。

【書類名】 図面

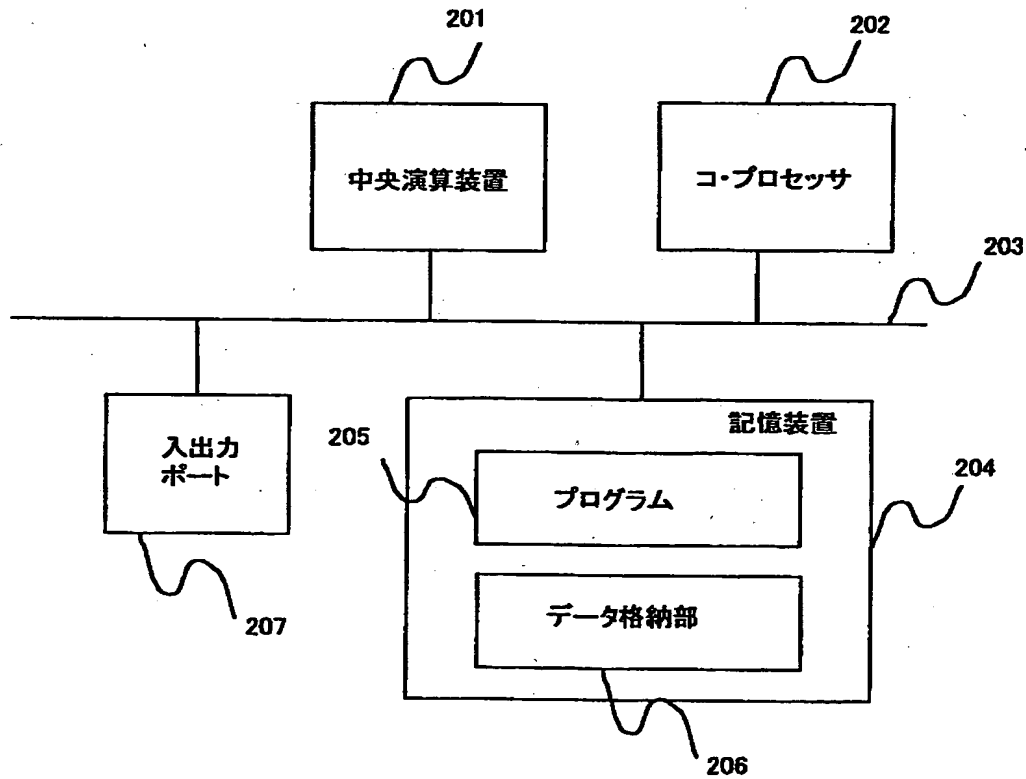
【図 1】

図 1 ICカード



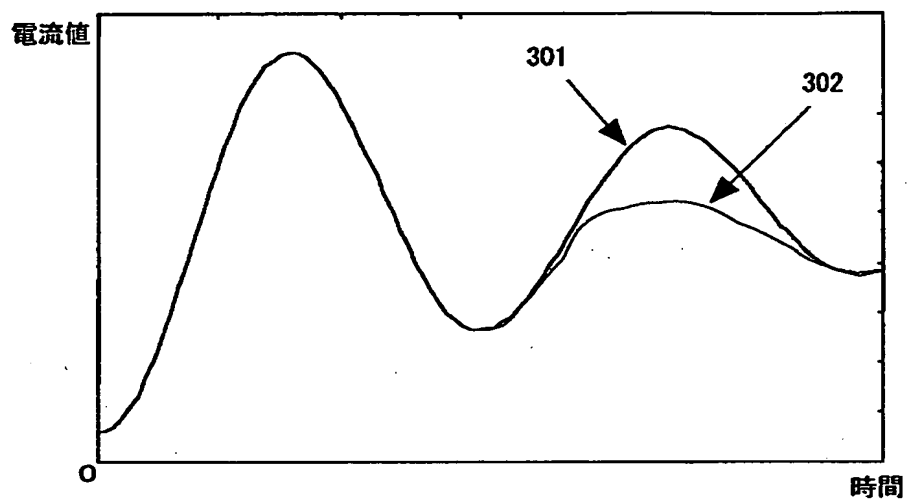
【図 2】

図 2 マイクロコンピュータの構成



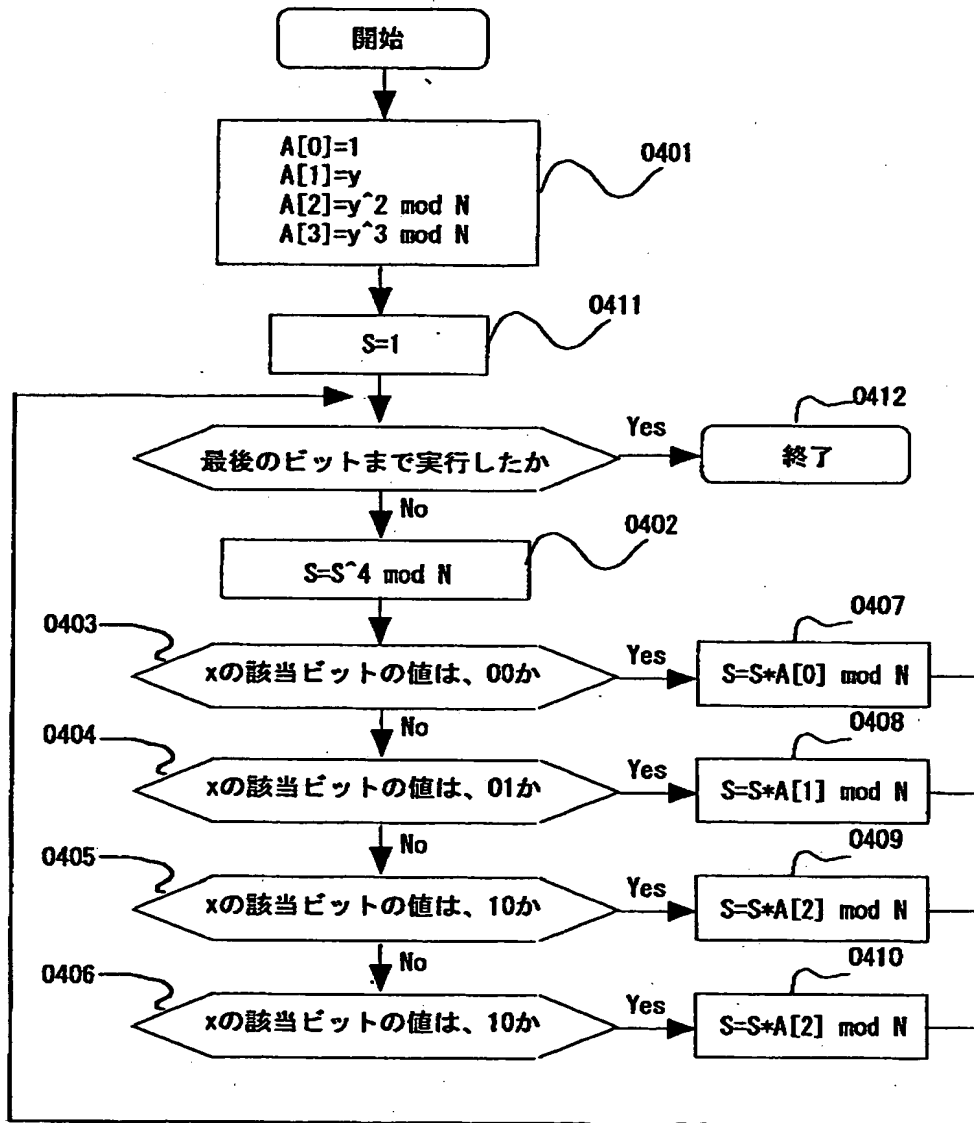
【図3】

図3 消費電流波形



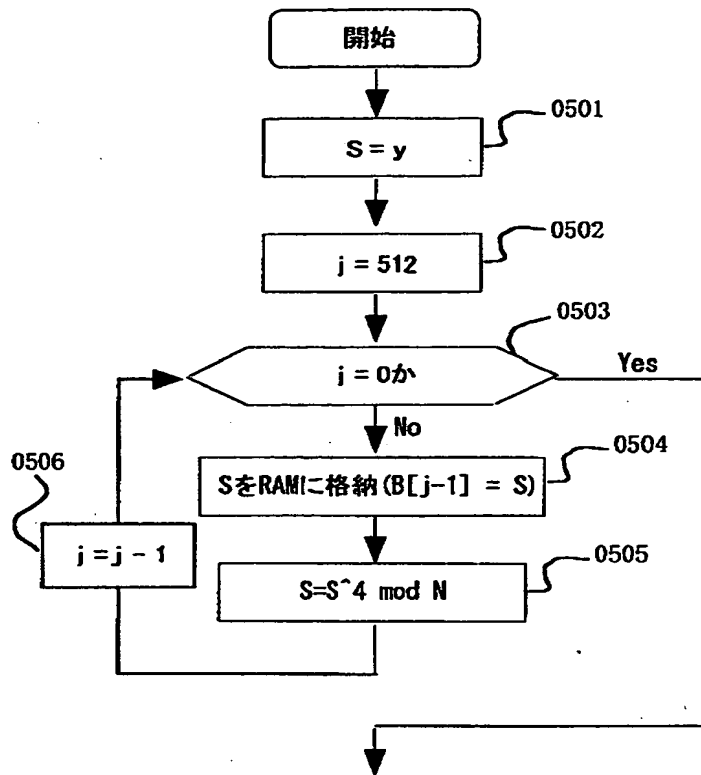
【図 4】

図 4 アディション・チェイン方式を用いたべき乗剰余計算



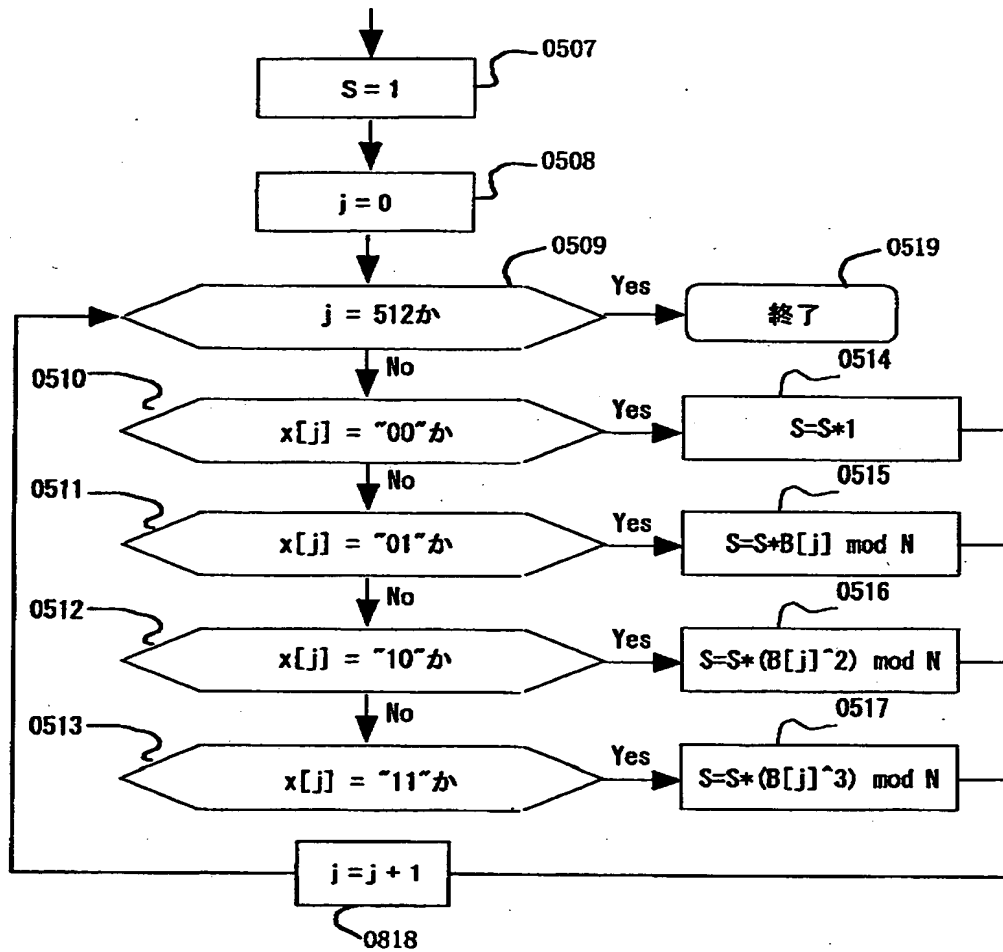
【図 5】

図 5 アディション・チェイン方式を用いたべき乗剰余計算
【テーブル作成部】



【図 6】

図6 アディション・チェイン方式を用いたべき乗剰余計算
【モジュラ乗算部】



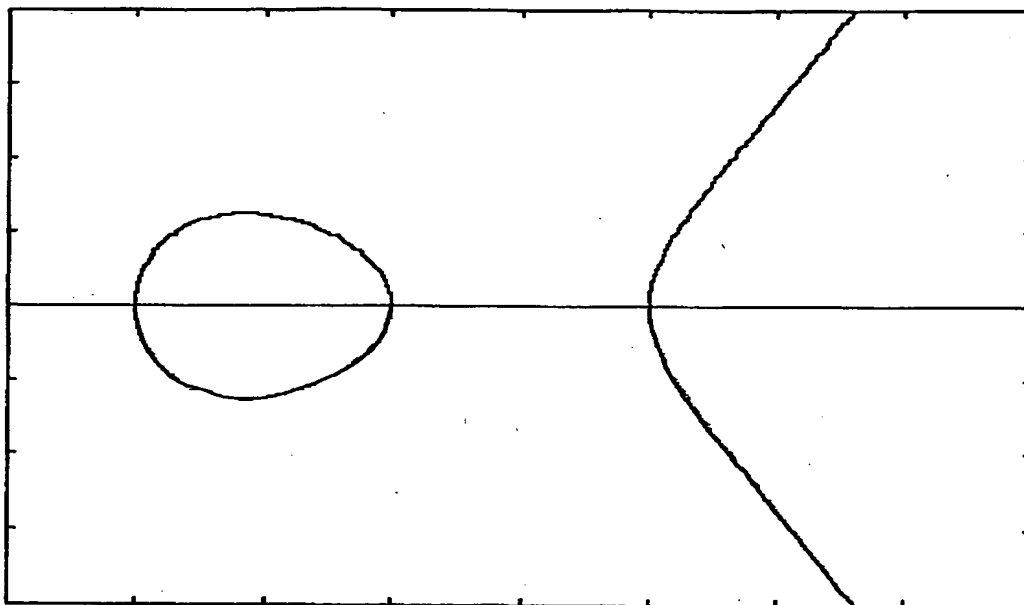
【図 7】

図 7 アディション・チェイン方式を用いたべき乗剰余計算
【テーブルのメモリ配置】

ADDR	B[0]
ADDR+128*1	B[1]
ADDR+128*2	B[2]
ADDR+128*3	B[3]
ADDR+128*4	B[4]
.....	
ADDR+128*509	B[509]
ADDR+128*510	B[510]
ADDR+128*511	B[511]

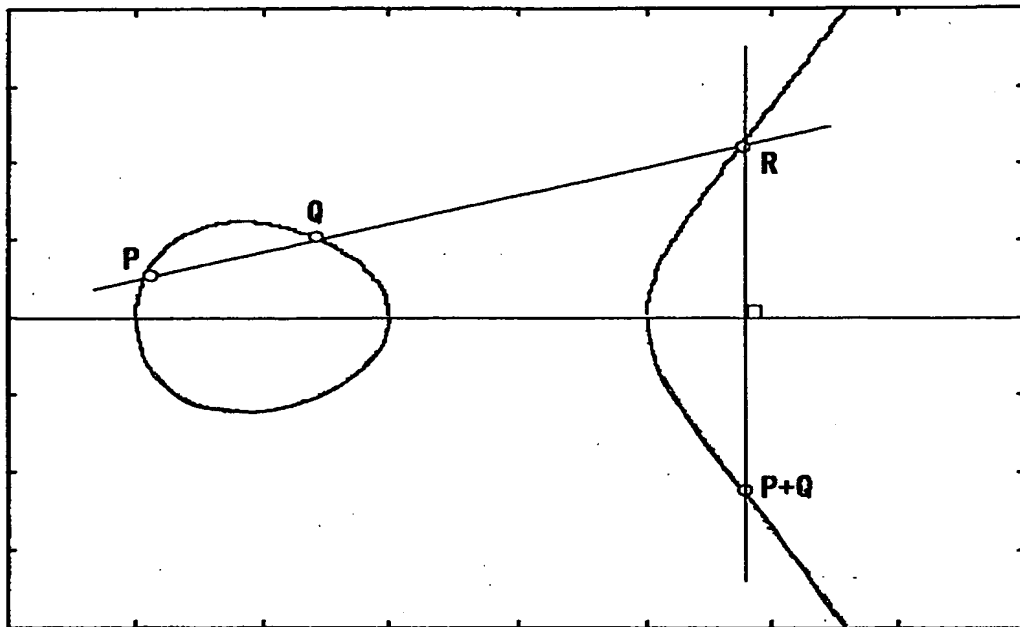
【図 8】

図 8 楕円曲線の形状



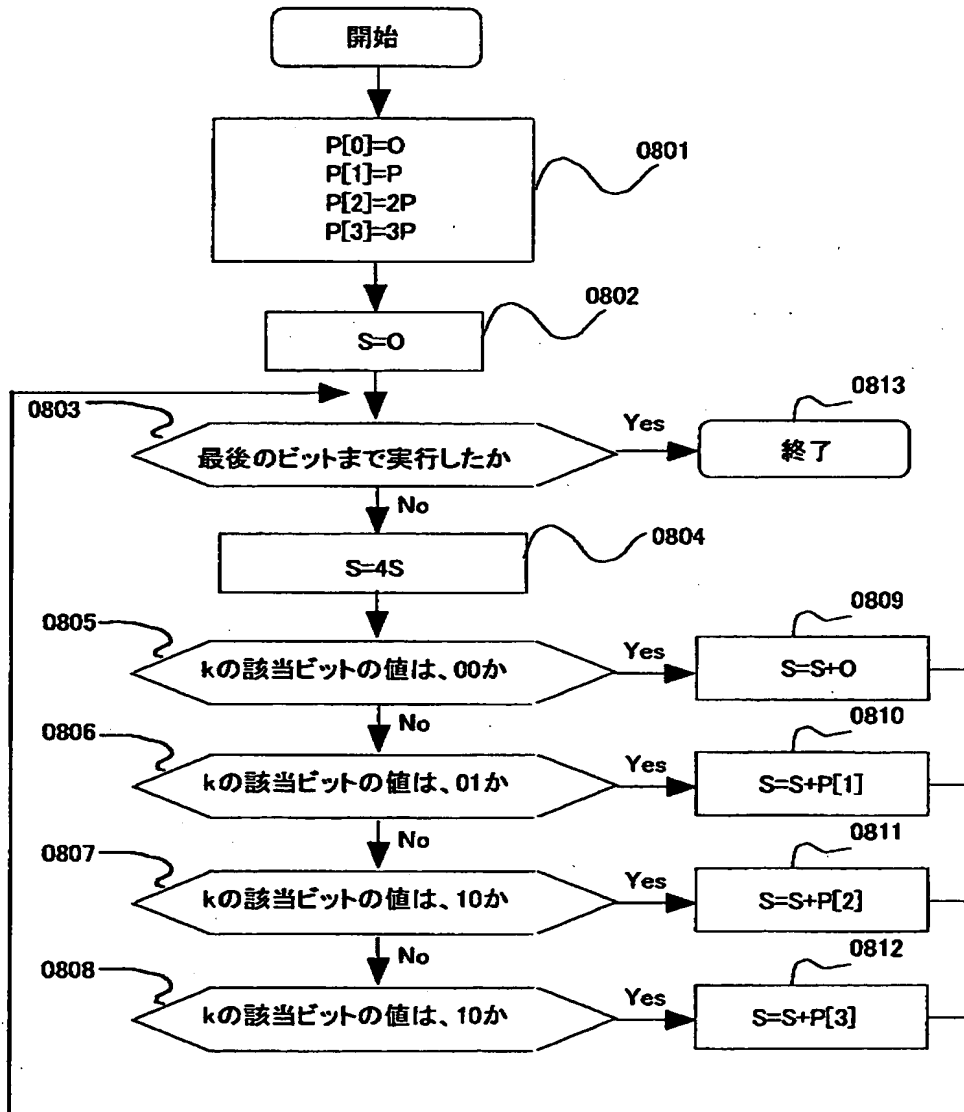
【図9】

図9 楕円曲線上の加法



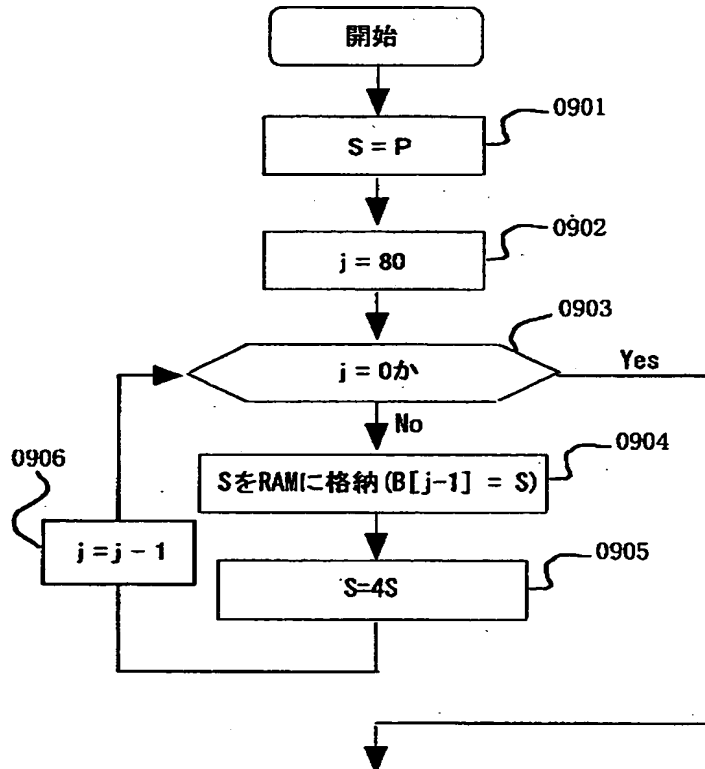
【図10】

図10 アディション・チェーン方式を用いたスカラー倍計算



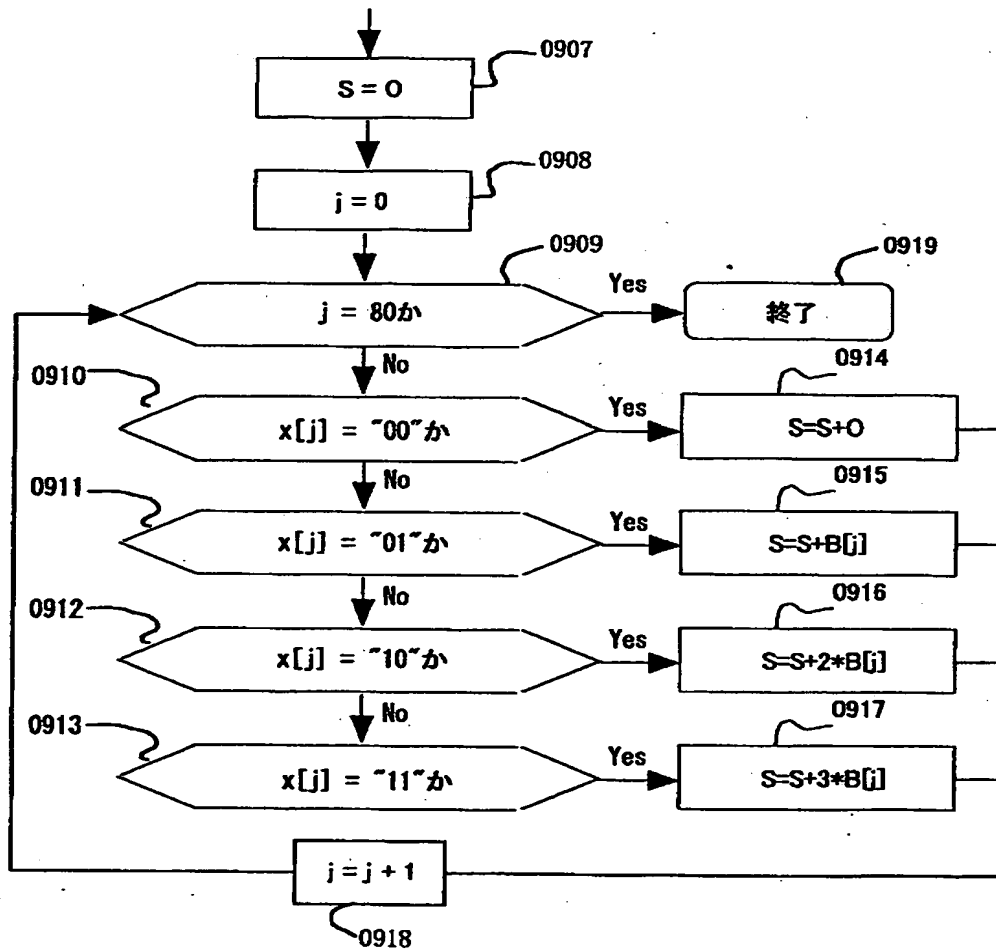
【図 11】

図 11 楕円曲線上のスカラ乗計算部
【テーブル作成部】



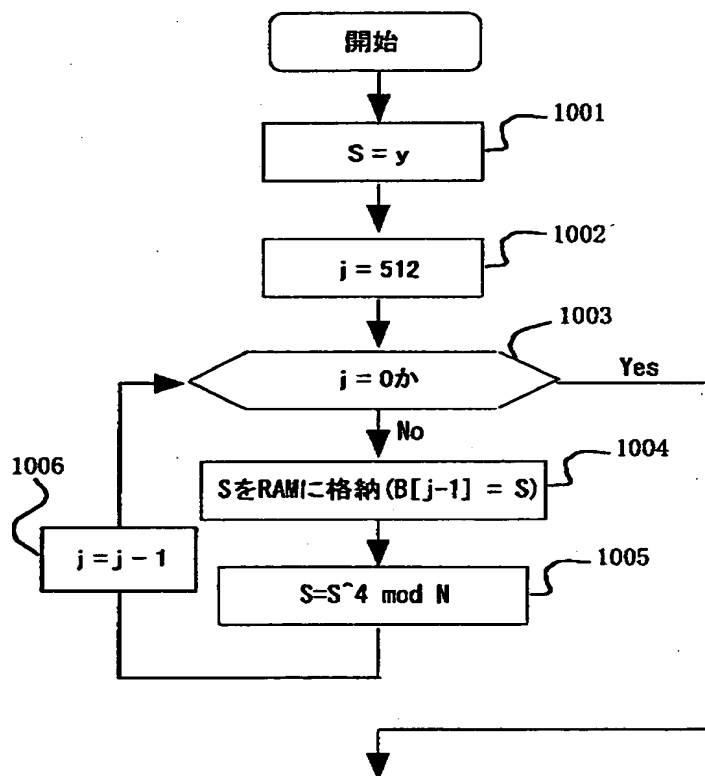
【図 12】

図12 楕円曲線上のスカラ乗計算部
【スカラ依存処理】



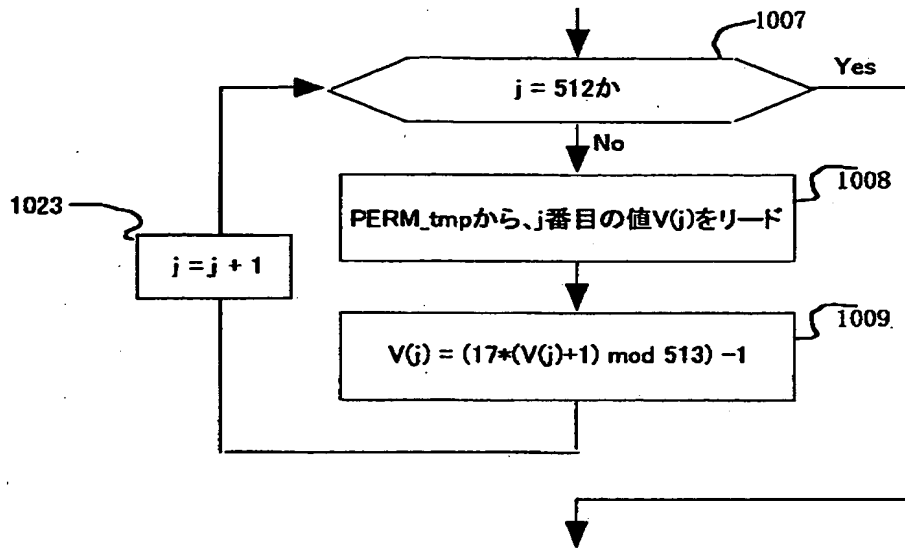
【図 13】

図13 RSA暗号処理に対する本発明実施例
【テーブル作成部】



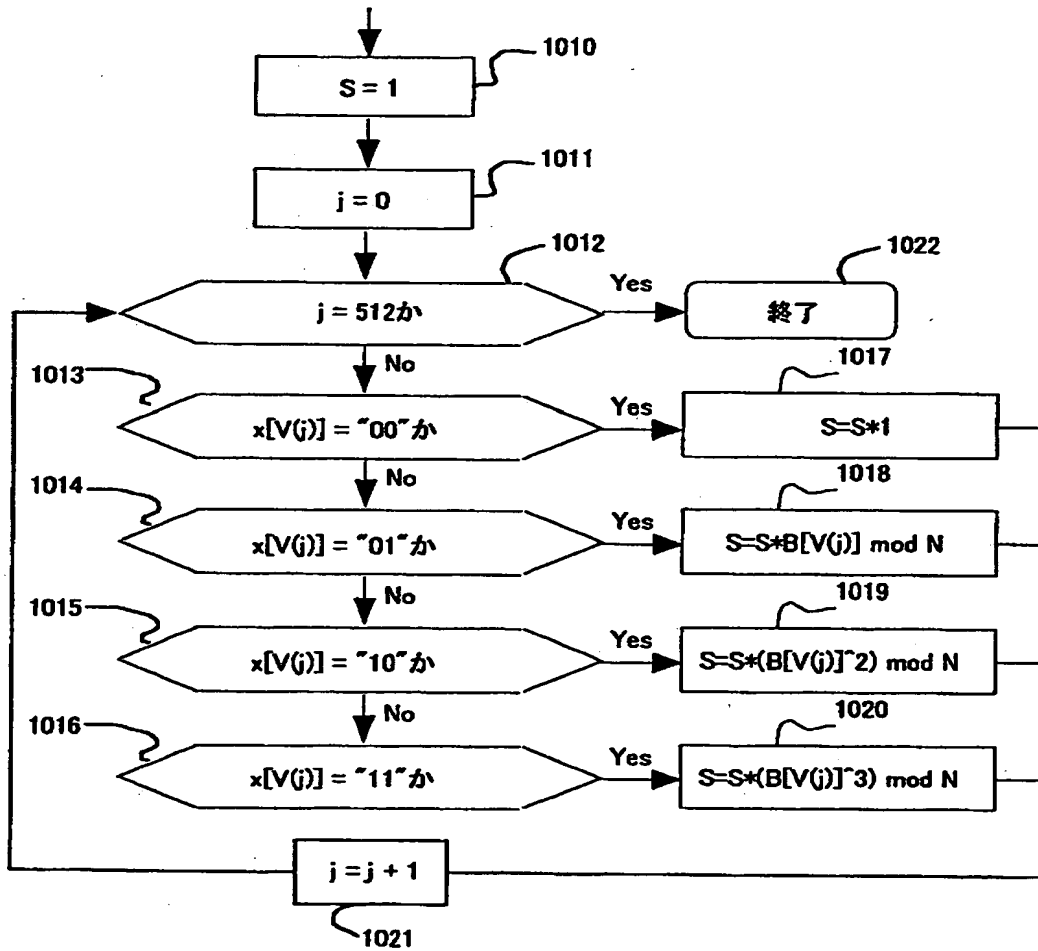
【図 1 4】

図14 RSA暗号処理に対する本発明実施例
【ランダム置換生成部】



【図 15】

図15 RSA暗号処理に対する本発明実施例
【モジュラ乗算部】



【図 1 6】

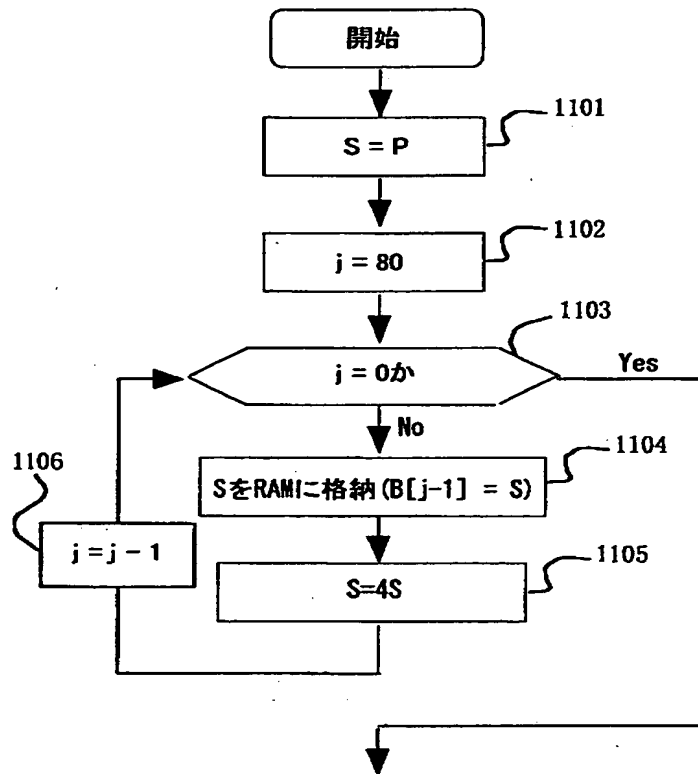
図16 RSA暗号処理に対する本発明実施例
【置換用テーブルPERM_tmp】

ADDR	127	V(0)
ADDR+2*1	21	V(1)
ADDR+2*2	170	V(2)
ADDR+2*3	509	V(3)
ADDR+2*4	342	V(4)

ADDR+2*509	263	V(509)
ADDR+2*510	428	V(510)
ADDR+2*511	79	V(511)

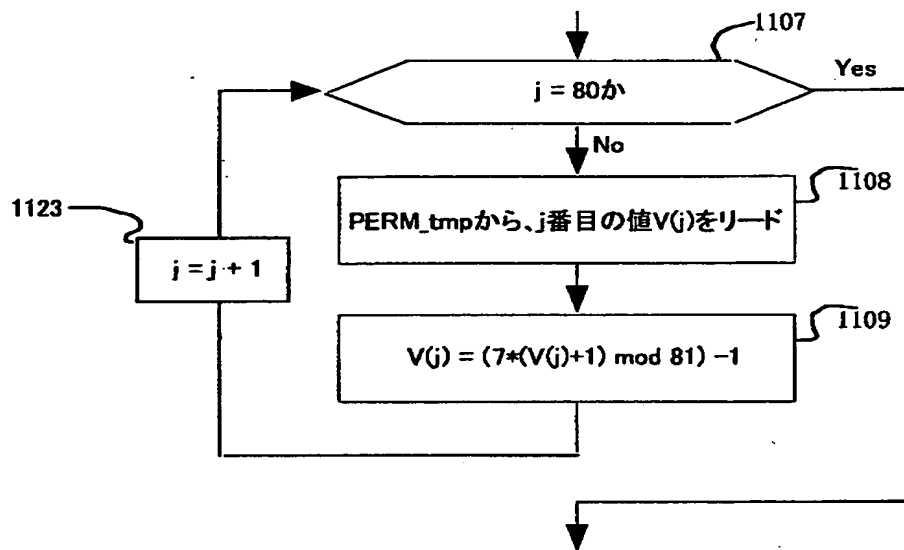
【図 17】

図17 楕円曲線暗号処理に対する本発明実施例
【テーブル作成部】



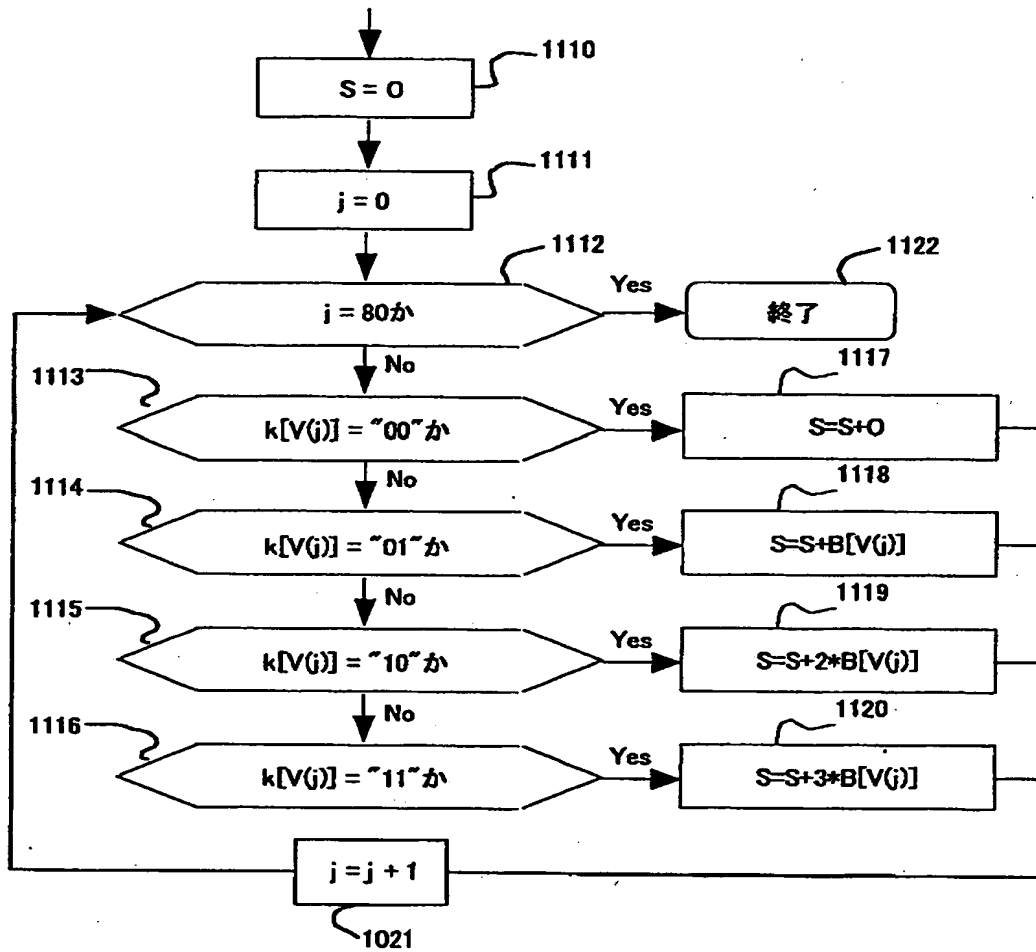
【図 1 8】

図18 楕円曲線暗号処理に対する本発明実施例
【ランダム置換生成部】



【図19】

図19 楕円曲線暗号処理に対する本発明実施例
【スカラー依存処理】



【図 2 0】

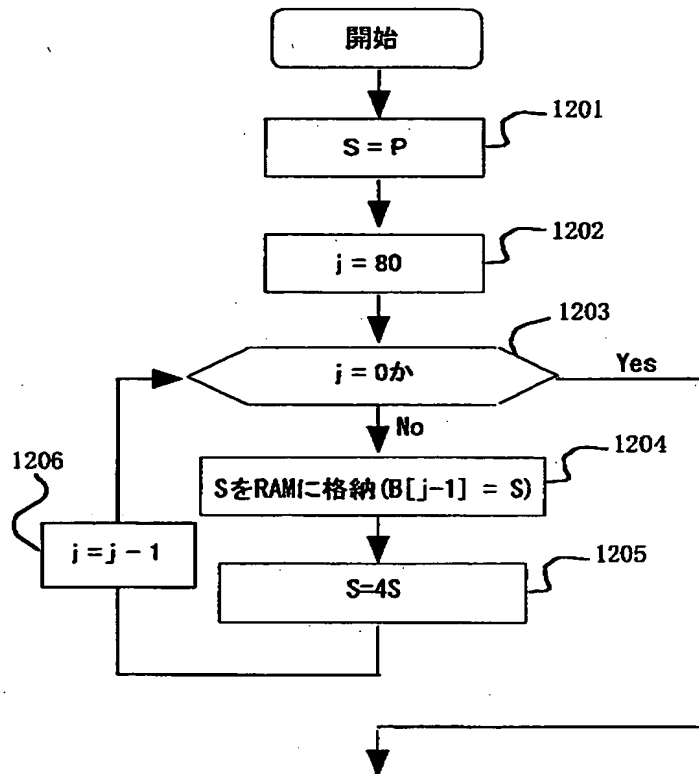
図20 楕円曲線暗号処理に対する本発明実施例
【置換用テーブルPERM_tmp】

ADDR	27	V(0)
ADDR+2*1	21	V(1)
ADDR+2*2	70	V(2)
ADDR+2*3	59	V(3)
ADDR+2*4	42	V(4)

ADDR+2*77	63	V(77)
ADDR+2*78	69	V(78)
ADDR+2*79	3	V(79)

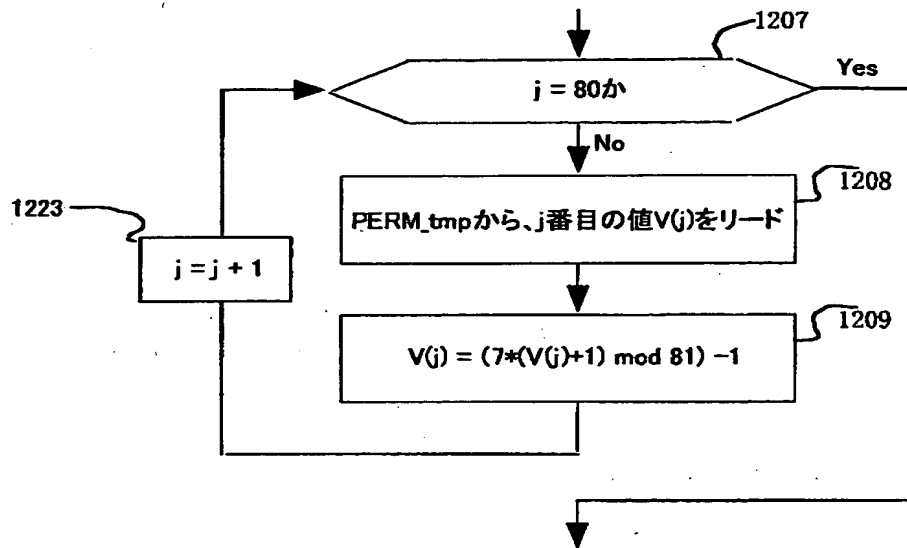
【図 21】

図21 楕円曲線暗号処理に対する本発明実施例(その2)
【テーブル作成部】



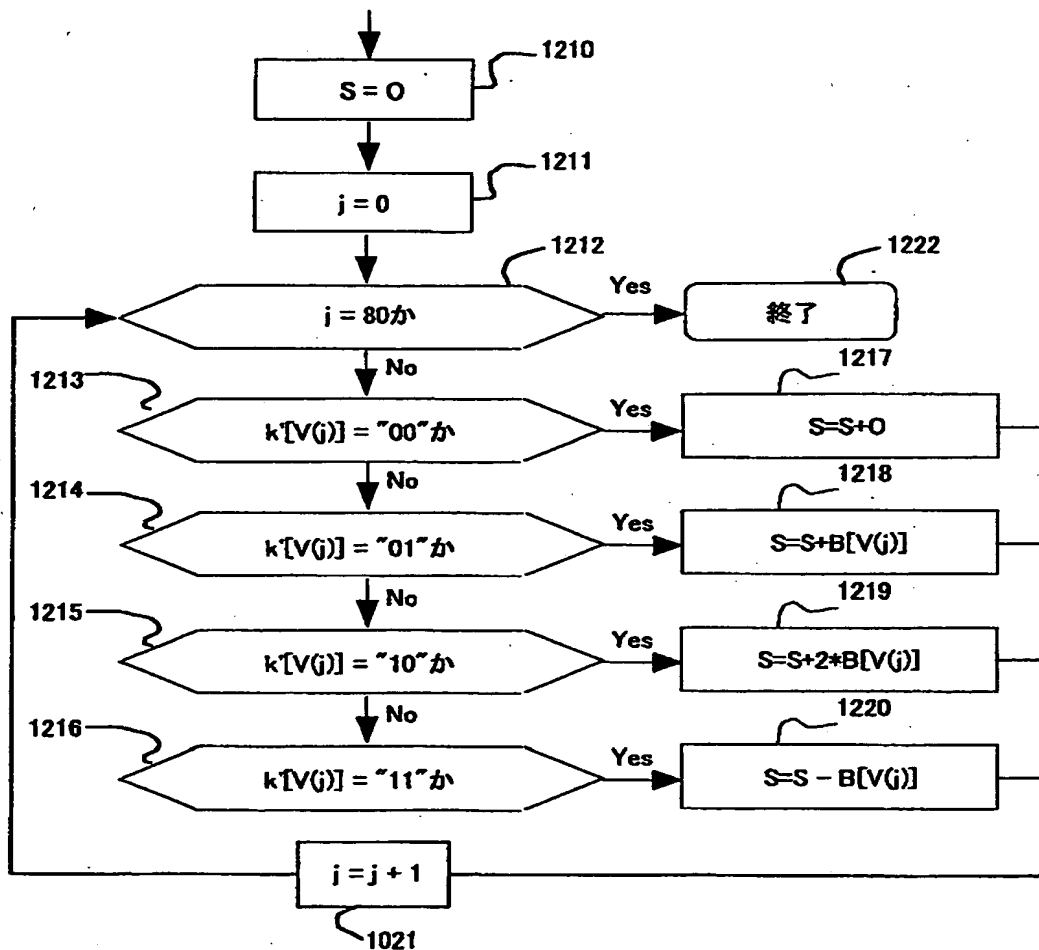
【図 22】

図22 楕円曲線暗号処理に対する本発明実施例(その2)
【ランダム置換生成部】



【図 23】

図23 楕円曲線暗号処理に対する本発明実施例(その2)
【スカラー依存処理】



【書類名】 要約書

【要約】

【課題】 ICカードチップでの処理順序をアタッカーに推定されないように変更することにより、消費電力の波形から、処理や暗号鍵の推測を困難にすることにある。

【解決手段】 プログラムを格納するプログラム格納部およびデータを格納するデータ格納部を有する記憶手段と、演算処理装置と、前記演算処理装置で演算処理する対象となるデータを入力する手段および前記データの演算処理装置での演算処理結果を出力する手段とを有する情報処理装置内で2つの整数 K_1 、 K_2 に対して $F(K_1+K_2, A)=F(K_1, A) \circ F(K_2, A)$ を満たす関数 F (ここで \circ は可換半群 S における二項演算を表す。 K は整数、 A は S の元を表す。)の値 $F(K, A)$ を求めるに際し、前記 K を m 個の整数の和 $K[0] + K[1] + \dots + K[m-1]$ に分解し、

前記 m 個の整数列 $0, 1, \dots, m-1$ の順序を置換 T で並べ替えた結果である $T(0), T(1), \dots, T(m-1)$ (これらの結果は前記整数列 $0, 1, \dots, m-1$ と一対一に対応するものとする。)を用いて

$F(K, A)=F(K[T(0)], A) \circ F(K[T(1)], A) \circ \dots \circ F(K[T(m-1)], A) \dots$ (式1)

の右辺中の項 $F(K[T(0)], A)$ から $F(K[T(m-1)], A)$ までを $F(K[T(0)], A), F(K[T(1)], A), \dots, F(K[T(m-1)], A)$ の順序で演算して $F(K, A)$ を算出することを特徴とする情報処理装置の演算方法にある。

【選択図】 図13

特2001-097964

認定・付加情報

特許出願の番号	特願2001-097964
受付番号	50100466192
書類名	特許願
担当官	第七担当上席 0096
作成日	平成13年 4月 2日

<認定情報・付加情報>

【提出日】	平成13年 3月30日
-------	-------------

次頁無

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所